@gwenshap gwen@confluent.io

What's this Stream Processing stuff anyway?

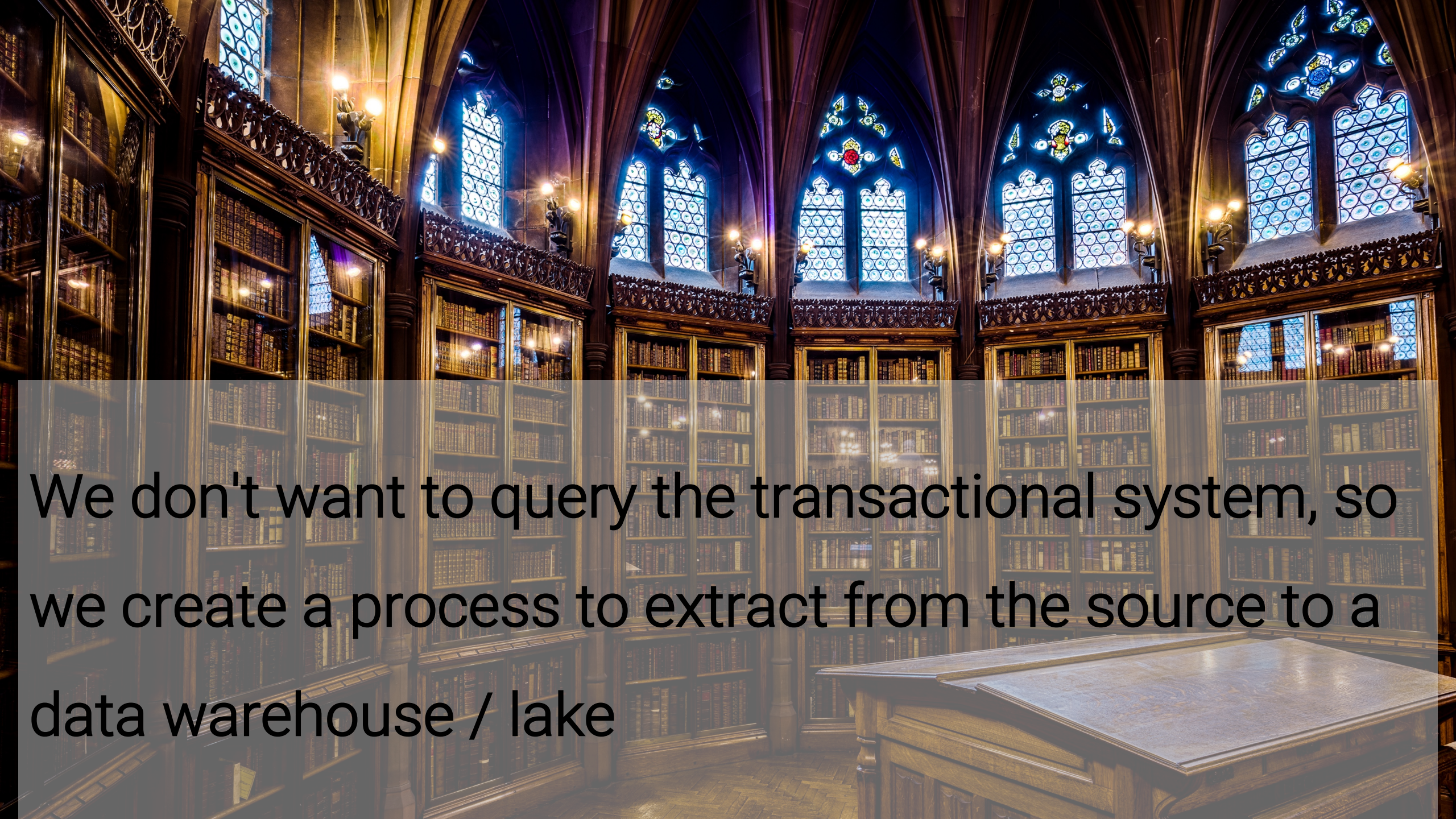
Oak Table World 2017

Gwen Shapira & Robin Moffatt

Confluent

@rmoff robin@confluent.io

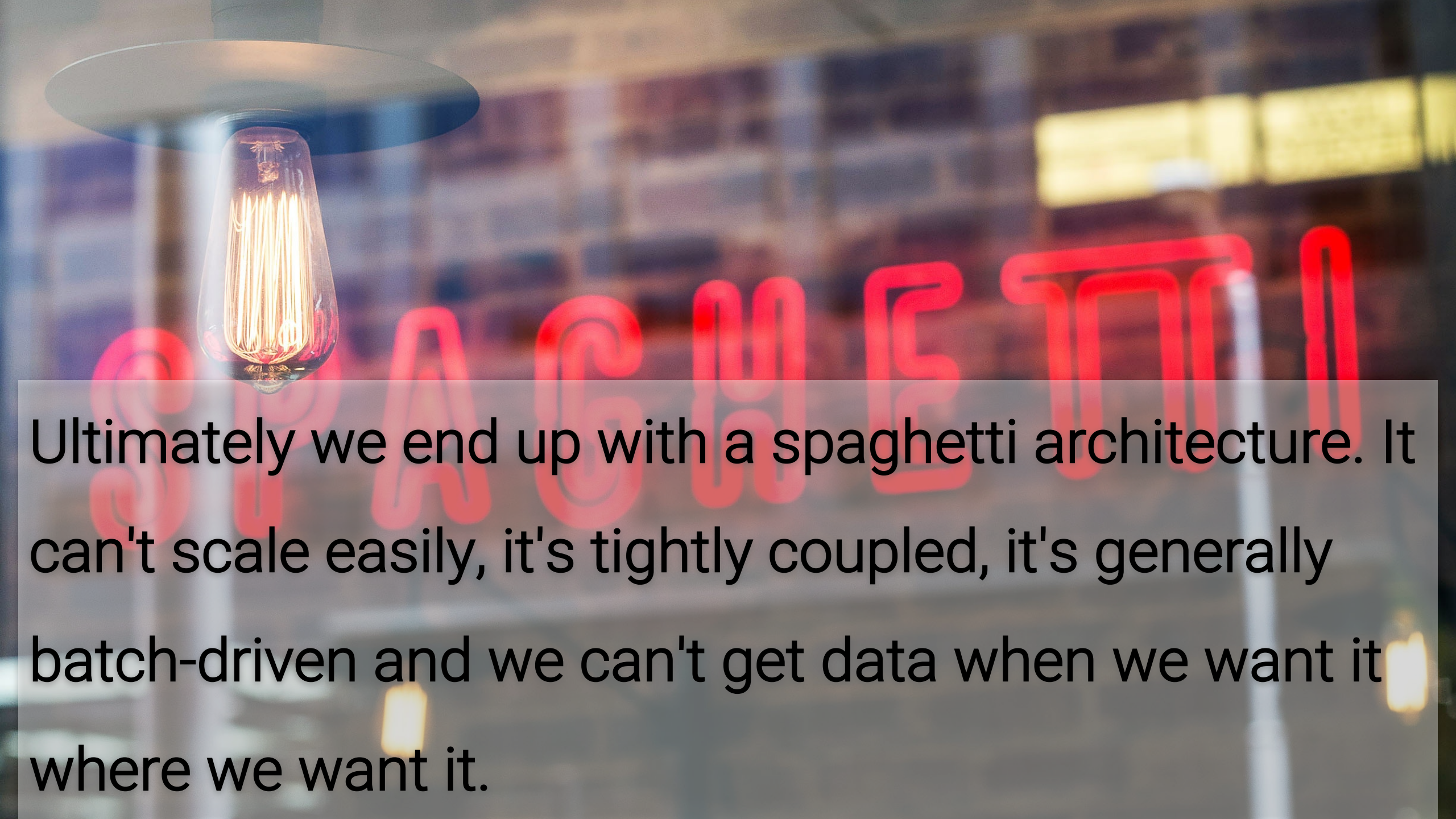
Let’s take a trip back in time. Each application has its own database for storing information. But we want that information elsewhere for analytics and reporting.



We don't want to query the transactional system, so we create a process to extract from the source to a data warehouse / lake

We want to unify data from multiple systems, so create conformed dimensions and batch processes to federate our data. This is all batch driven, so latency is built in by design.

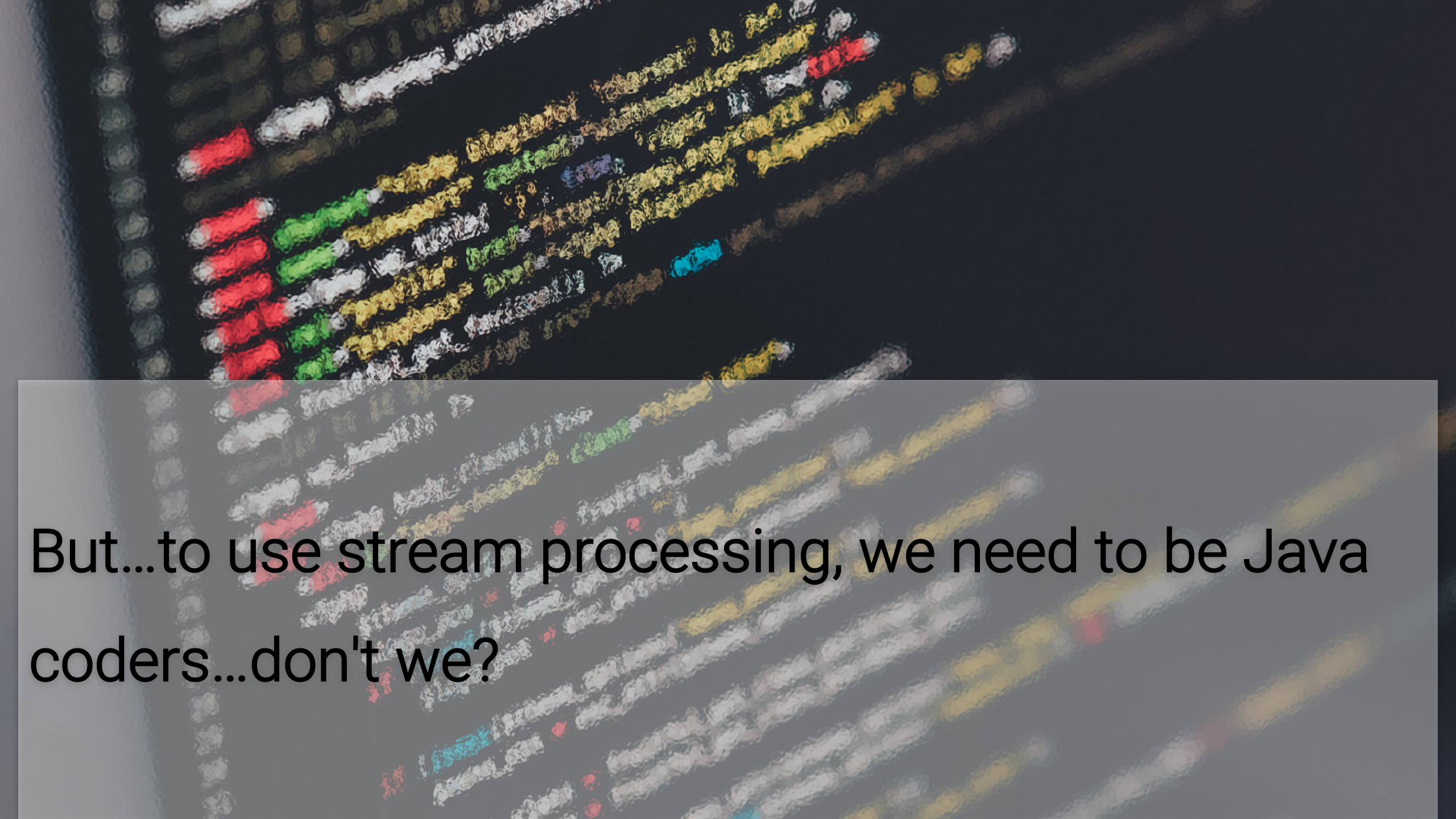
As well as our data warehouse, we want to use our transactional data to populate search replicas, Graph databases, noSQL stores...all introducing more point-to-point dependencies in our system



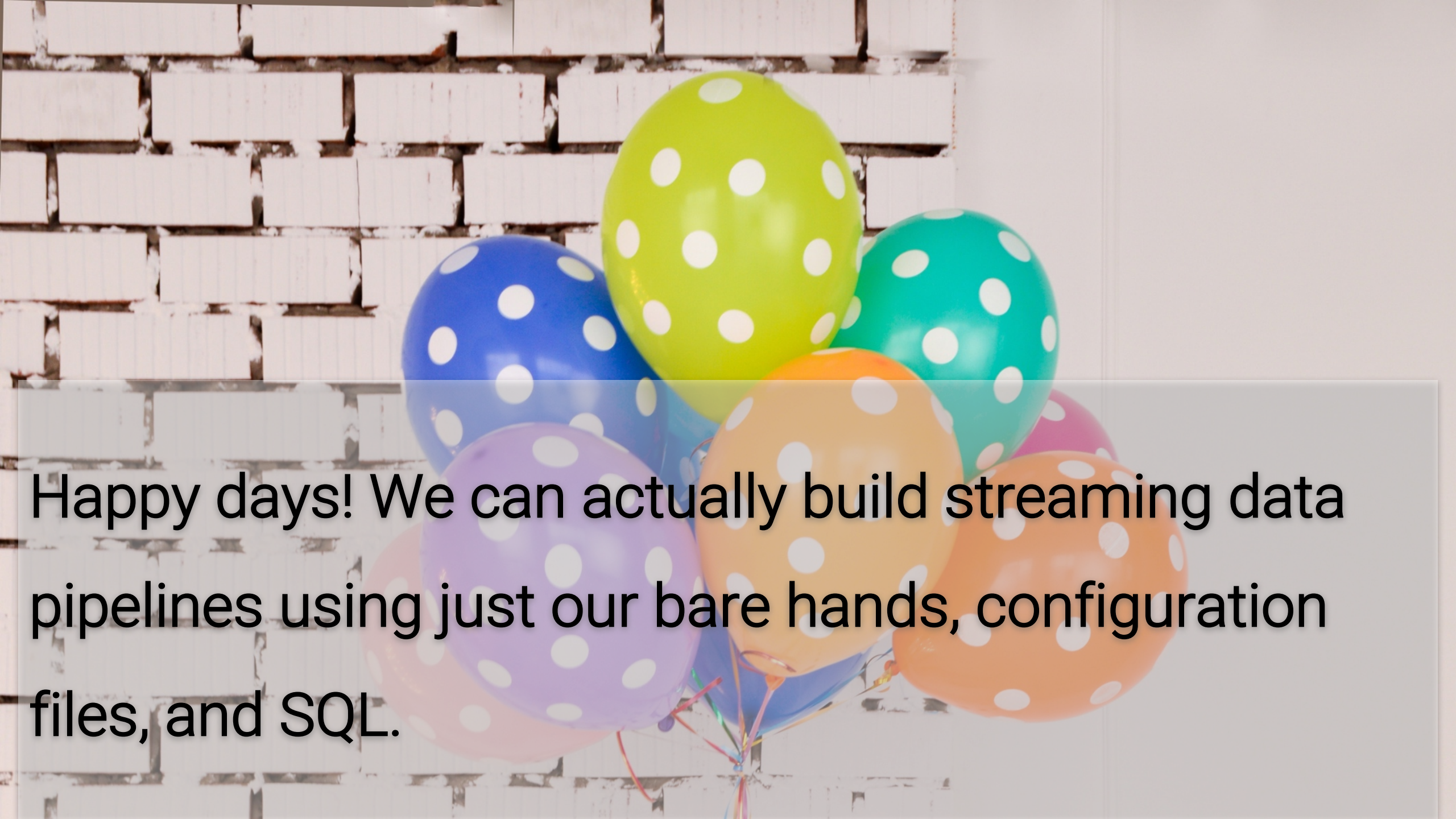
Ultimately we end up with a spaghetti architecture. It can't scale easily, it's tightly coupled, it's generally batch-driven and we can't get data when we want it where we want it.

But...there's hope!

Apache Kafka, a distributed streaming platform, enables us to decouple all our applications creating data from those utilising it. We can create low-latency streams of data, transformed as necessary.

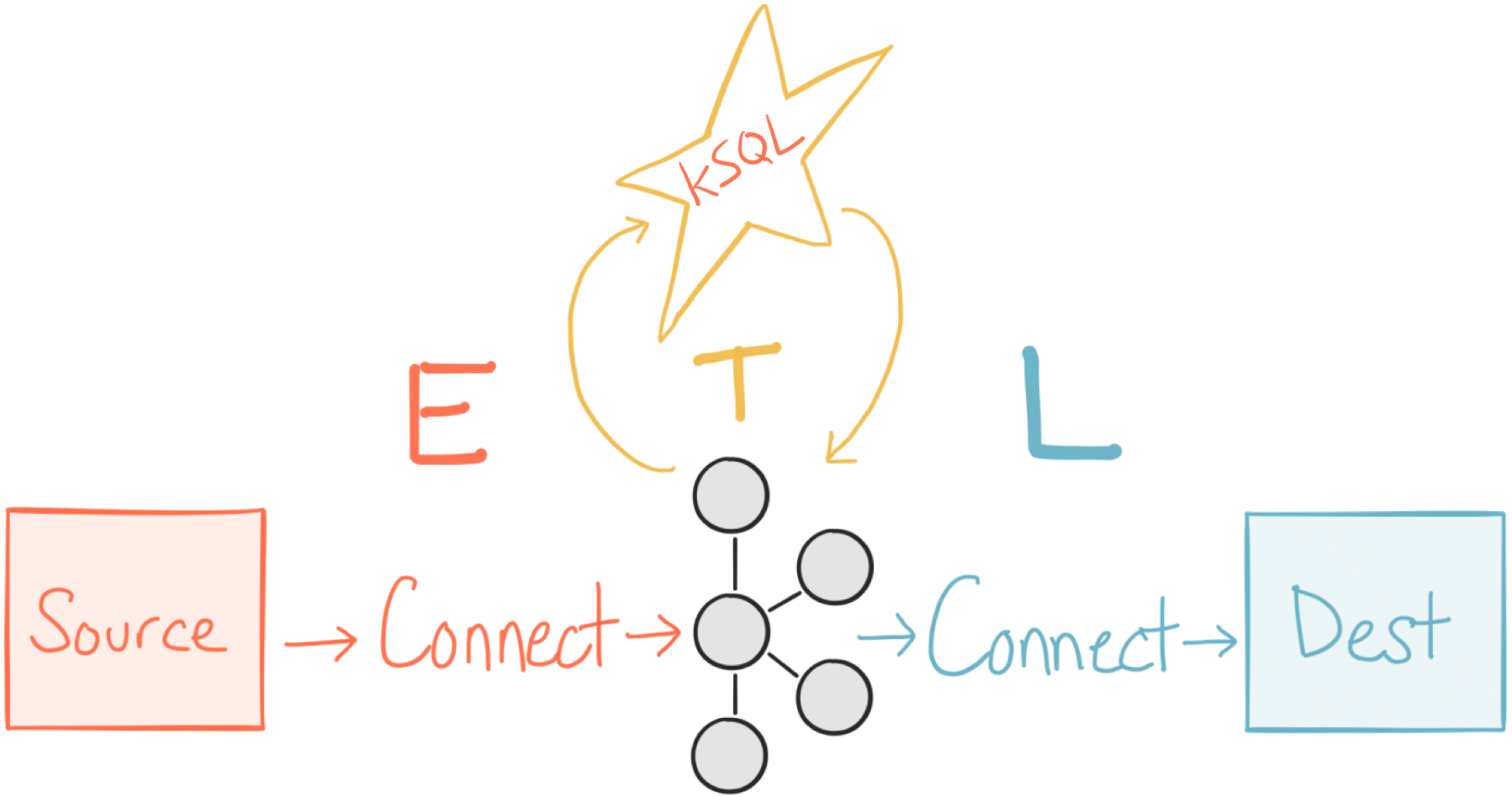


But...to use stream processing, we need to be Java coders...don't we?



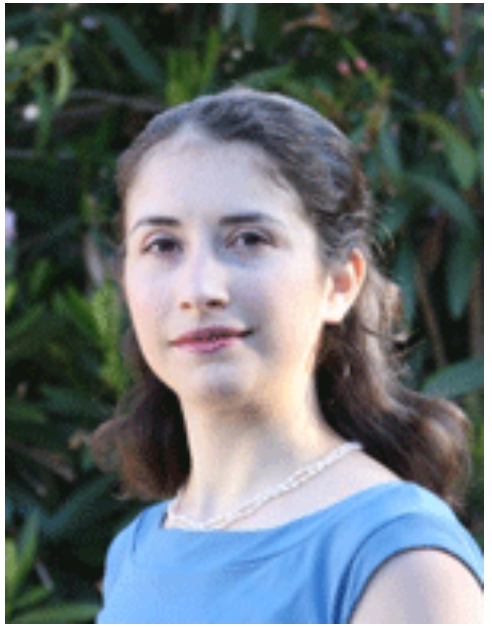
Happy days! We can actually build streaming data pipelines using just our bare hands, configuration files, and SQL.

Streaming ETL, with Apache Kafka and Confluent Platform



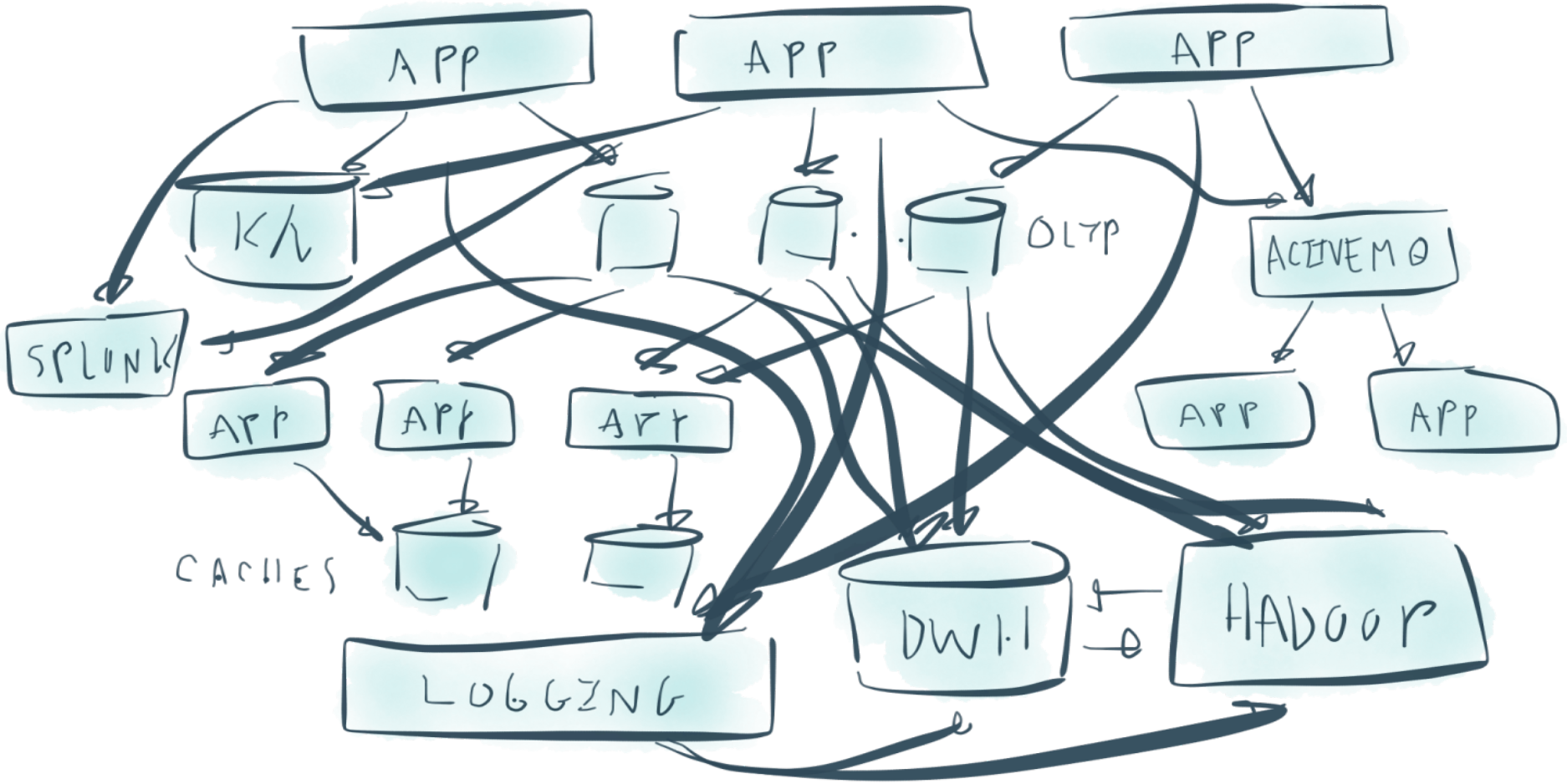
$ cat speakers.txt

- Gwen Shapira
  - Product Manager & Kafka Committer
  - @gwenshap

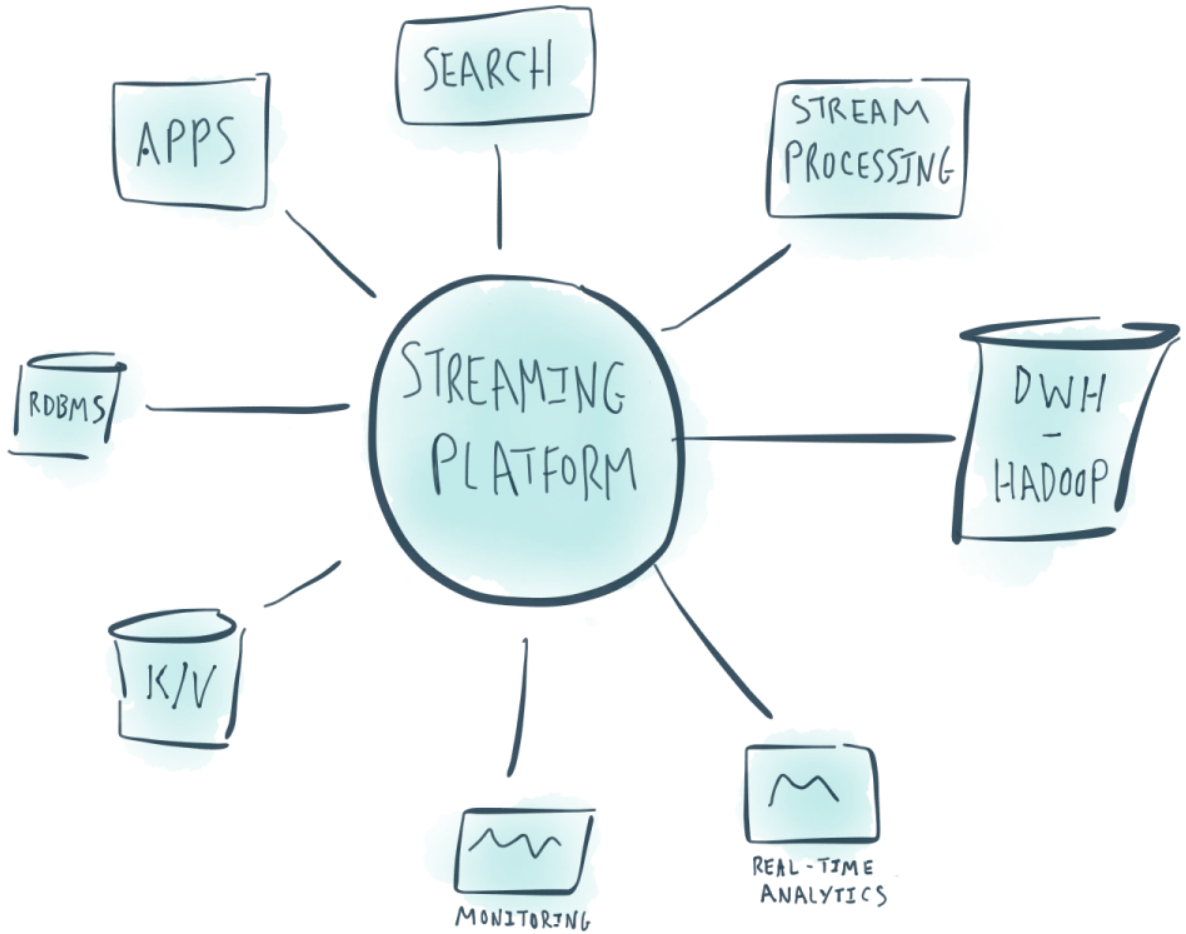


- Robin Moffatt
  - Partner Technology Evangelist @ Confluent
  - @rmoff

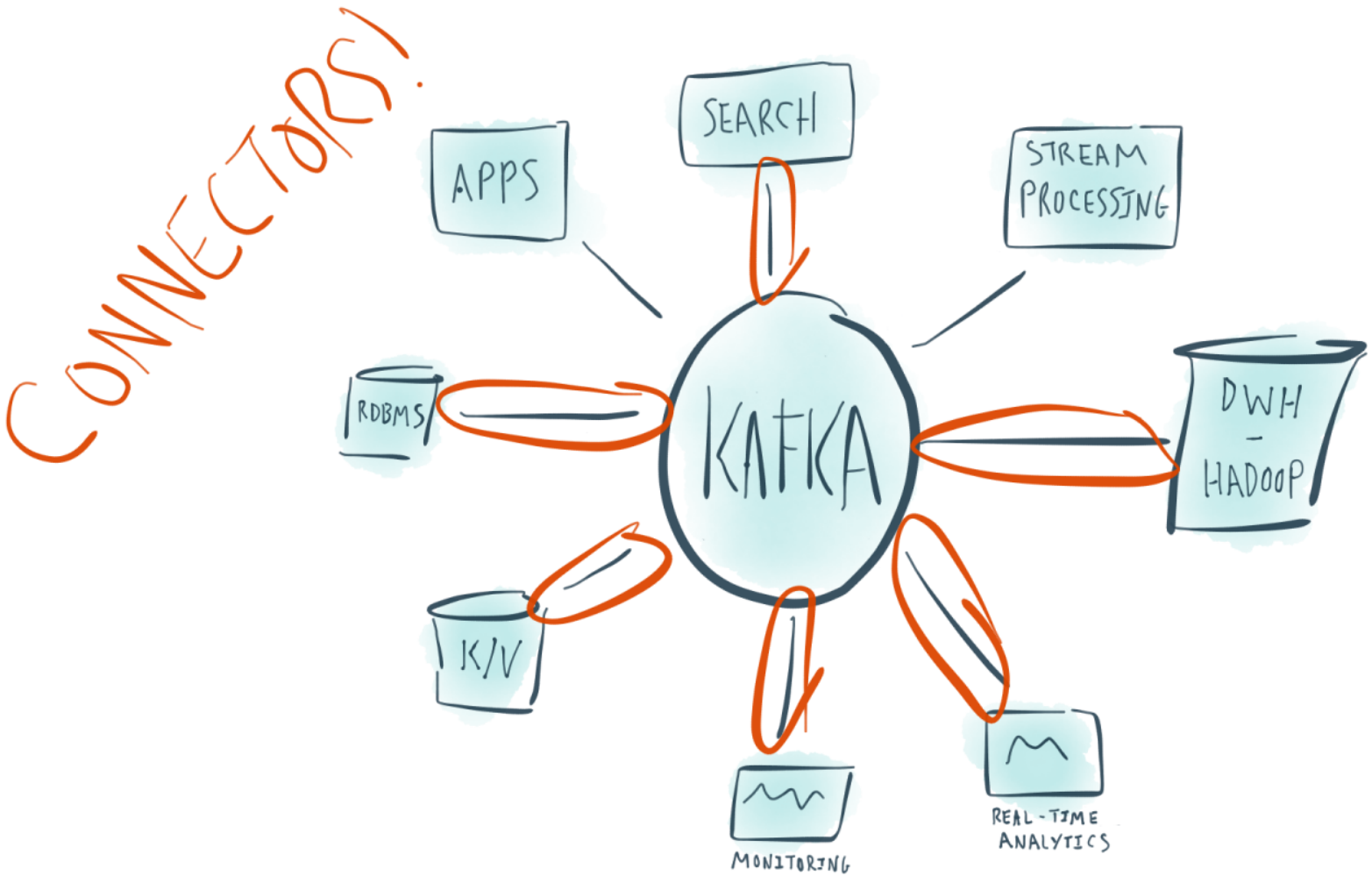
A GIANT MESS

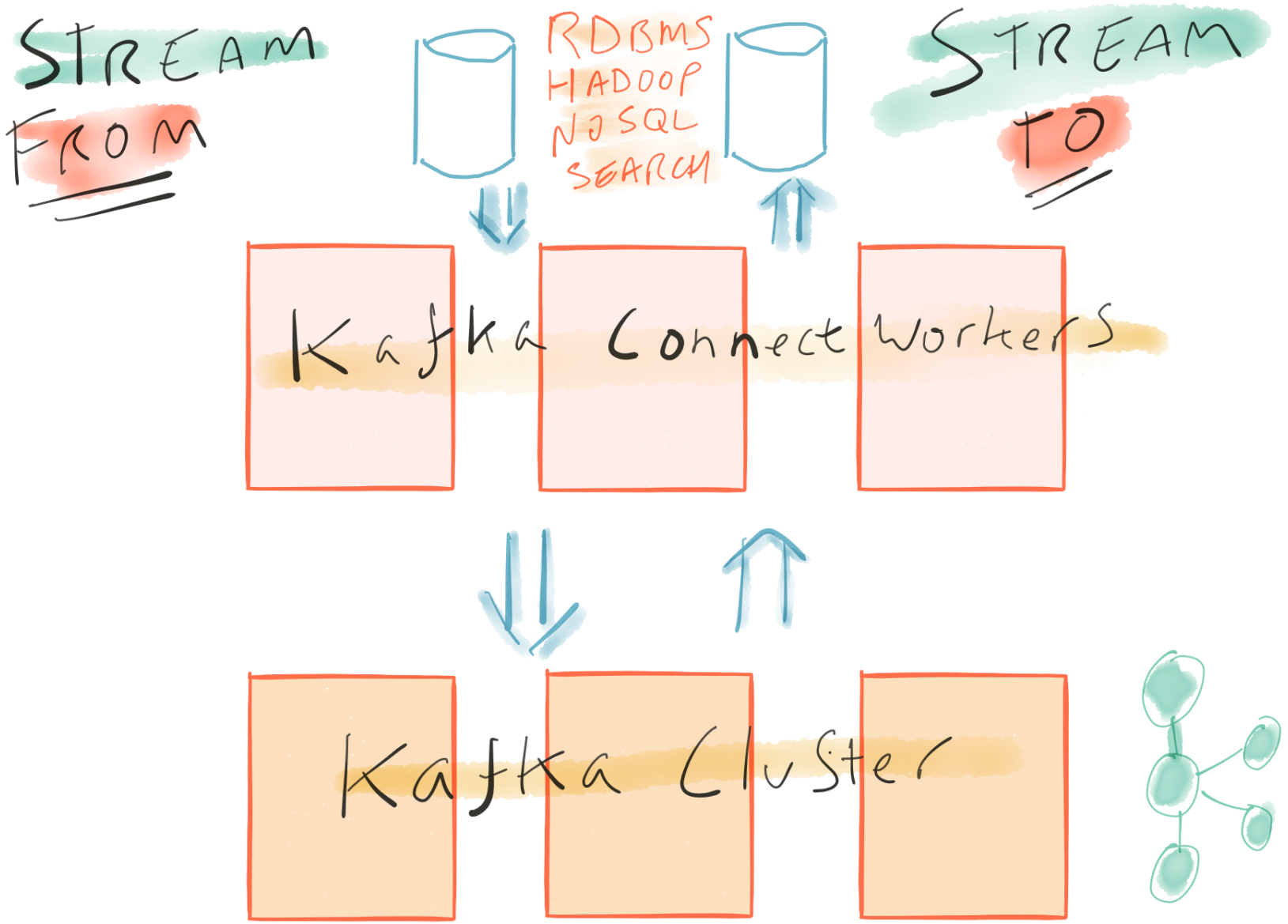

APP
APP
APP
K/V
OLTP
ACTIVEMQ
SPLUNK
APP
APP
APP
APP
APP
CACHES
DW1.1
HADOOP
LOGGING

STREAMING PLATFORM



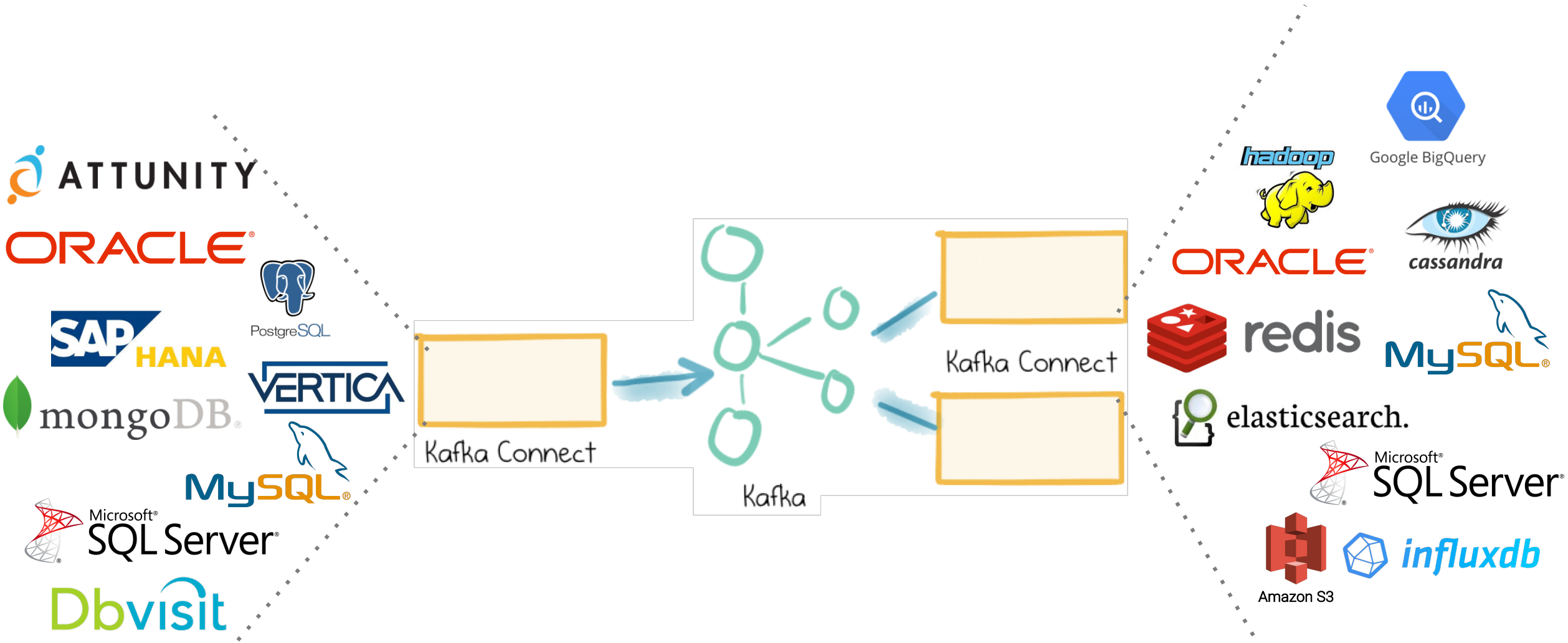
STREAMING PLATFORM





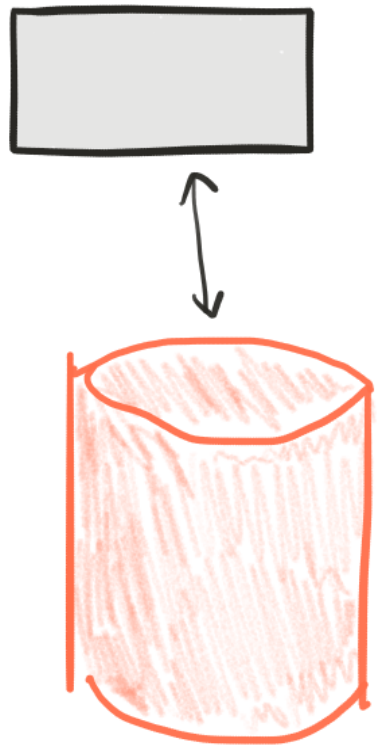
Build Data Pipelines

Kafka Connect : Stream data in and out of Kafka



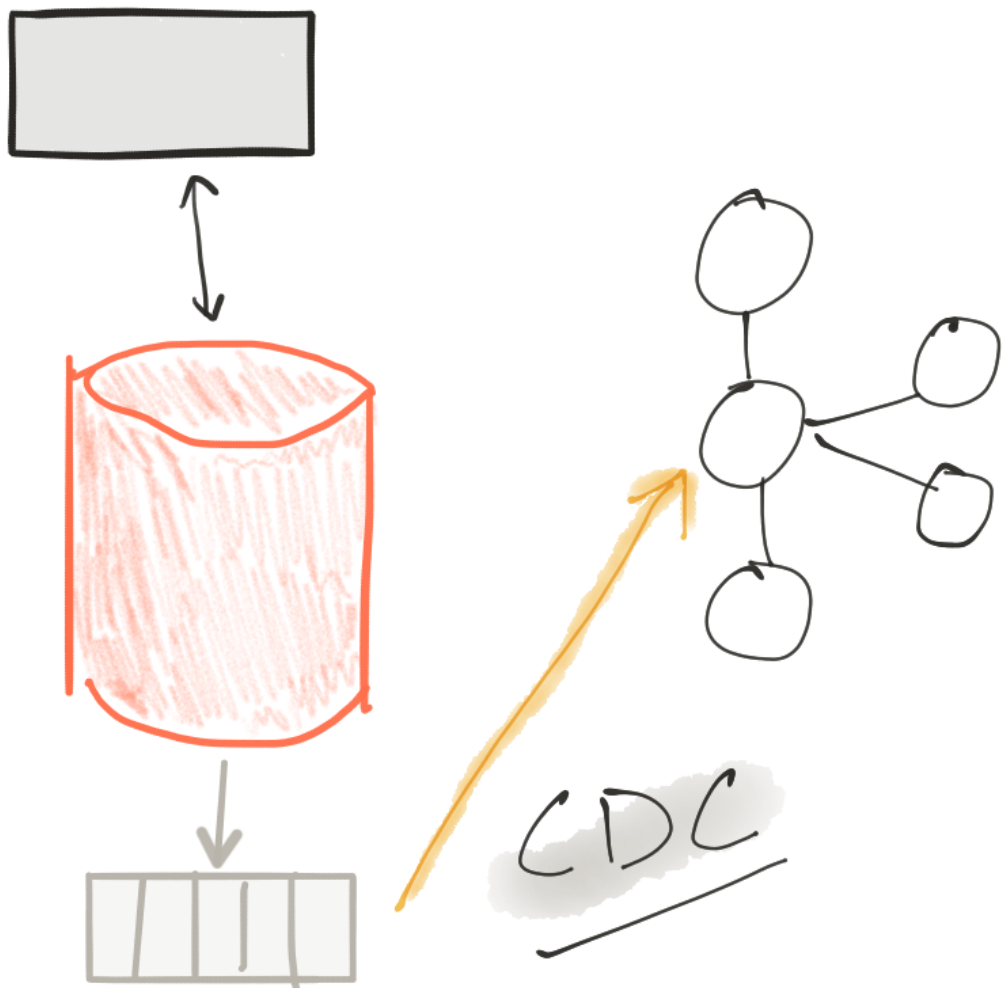
Streaming Application Data to Kafka

- Applications are rich source of events
- Modifying applications is not always possible or desirable
  - And what if the data gets changed within the database or by other apps?
- JDBC is one option for extracting data
  - Confluent Open Source includes JDBC source & sink connectors



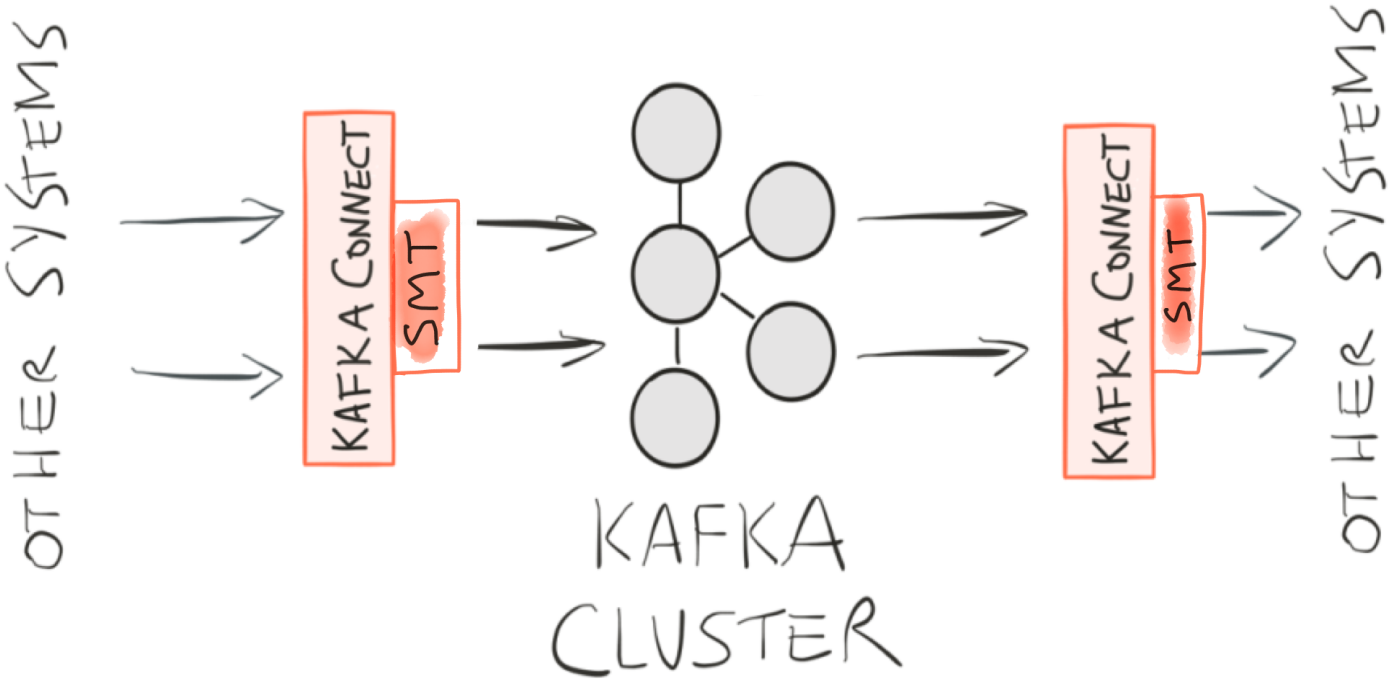
Liberate Application Data into Kafka with CDC

- Relational databases use transaction logs to ensure Durability of data
- Change-Data-Capture (CDC) mines the log to get raw events from the database
- CDC tools that integrate with Kafka Connect include:
  - Debezium
  - DBVisit
  - GoldenGate
  - Attunity
  - + more



Single Message Transform (SMT) -- Extract, **TRANSFORM**, Load…

- Modify events ***before storing*** in Kafka:
  - Mask/drop sensitive information
  - Set partitioning key
  - Store lineage
- Modify events ***going out*** of Kafka:
  - Route high priority events to faster data stores
  - Direct events to different Elasticsearch indexes
  - Cast data types to match destination

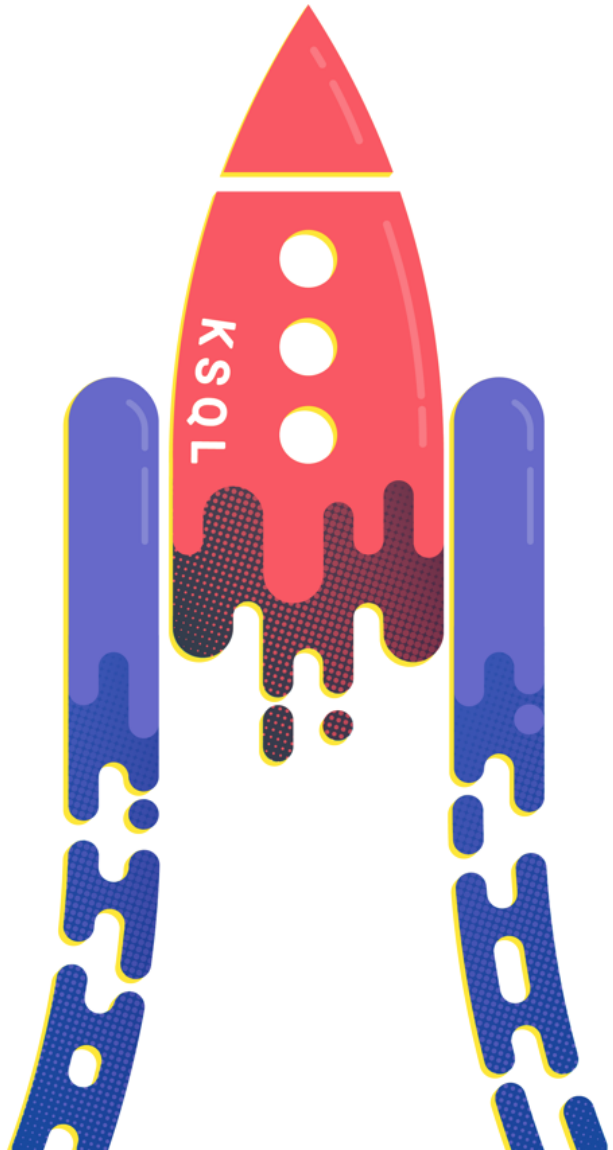


But I need to join...aggregate...filter...

A Developer Preview of

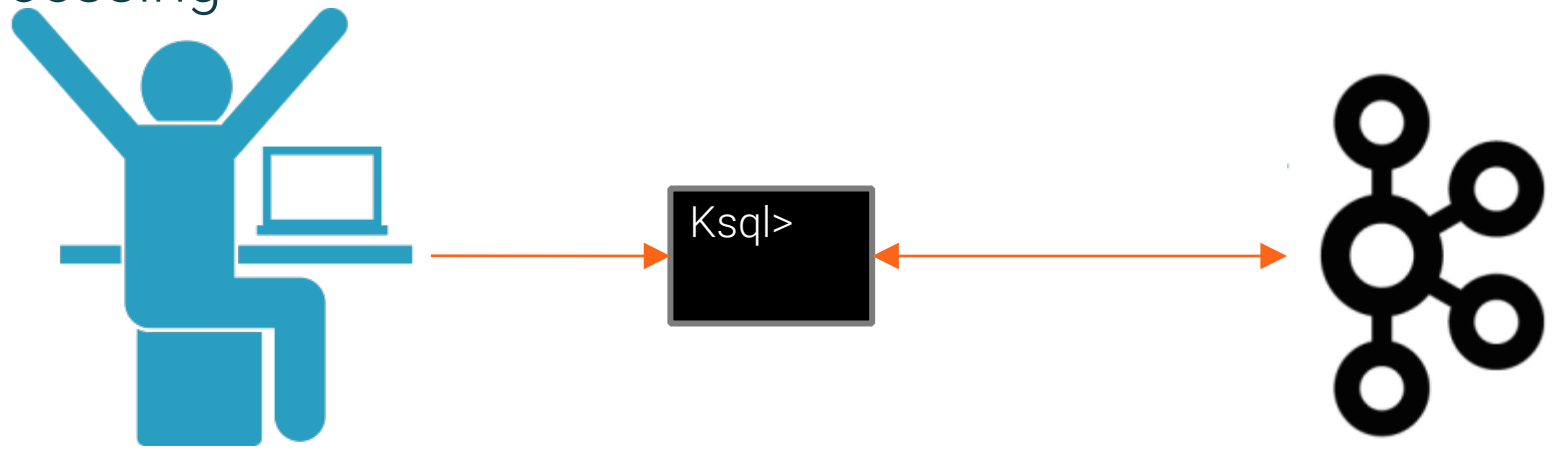
KSQL

An Open Source Streaming SQL Engine for Apache Kafka™



KSQL: a Streaming SQL Engine for Apache Kafka™ from Confluent

- Enables stream processing with zero coding required
- The simplest way to process streams of data in real-time
- Powered by Kafka: scalable, distributed, battle-tested
- All you need is Kafka–No complex deployments of bespoke systems for stream processing

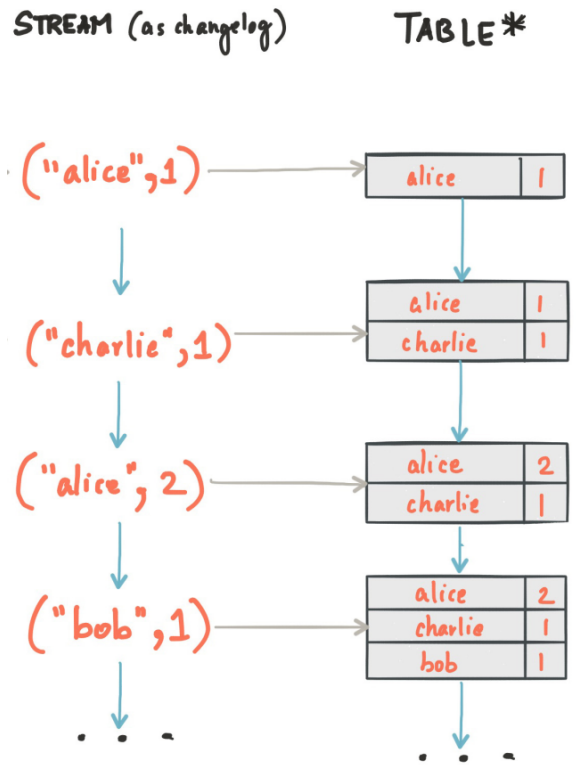


KSQL: the Simplest Way to Do Stream Processing

```
CREATE STREAM possible_fraud AS
SELECT card_number, count(*)
FROM authorization_attempts
WINDOW TUMBLING (SIZE 5 SECONDS)
GROUP BY card_number
HAVING count(*) > 3;
```

KSQL Concepts

- STREAM and TABLE as first-class citizens
  - Interpretations of topic content
- STREAM - data in motion
- TABLE - collected state of a stream
  - One record per key (per window)
  - Current values (compacted topic)
- STREAM – TABLE Joins



Window Aggregations

Three types supported (same as KStreams):

- TUMBLING: Fixed-size, non-overlapping, gap-less windows
  - `SELECT ip, count(*) AS hits FROM clickstream`

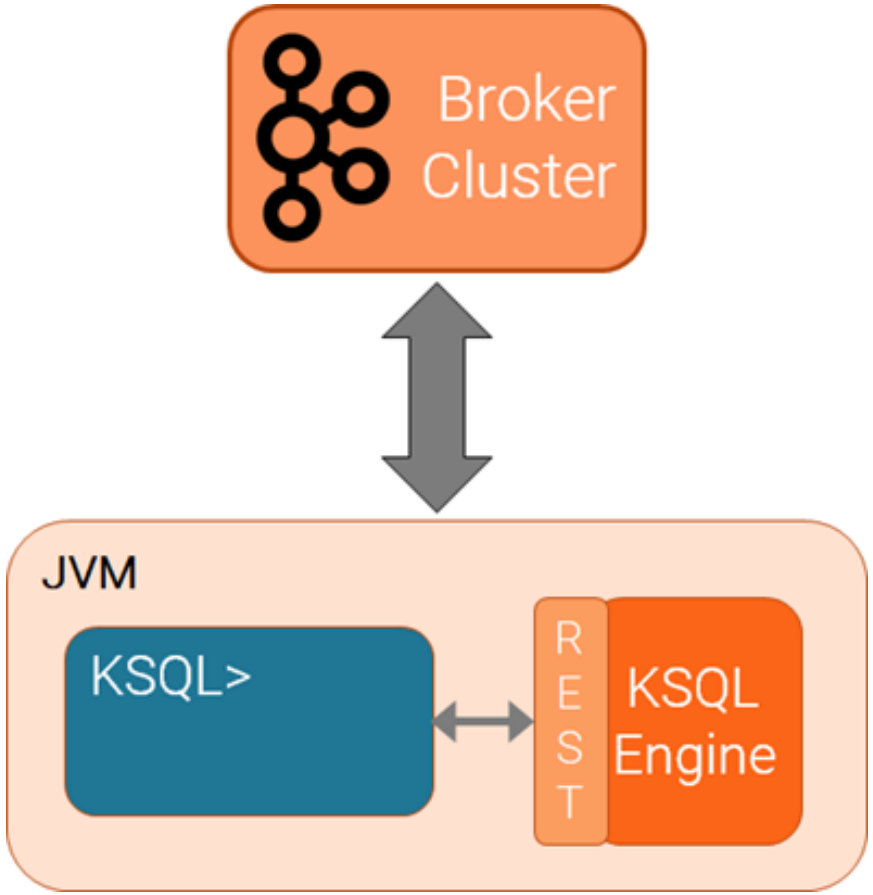
    `WINDOW TUMBLING (size 1 minute) GROUP BY ip;`
- HOPPING: Fixed-size, overlapping windows
  - `SELECT ip, SUM(bytes) AS bytes_per_ip_and_bucket FROM clickstream`

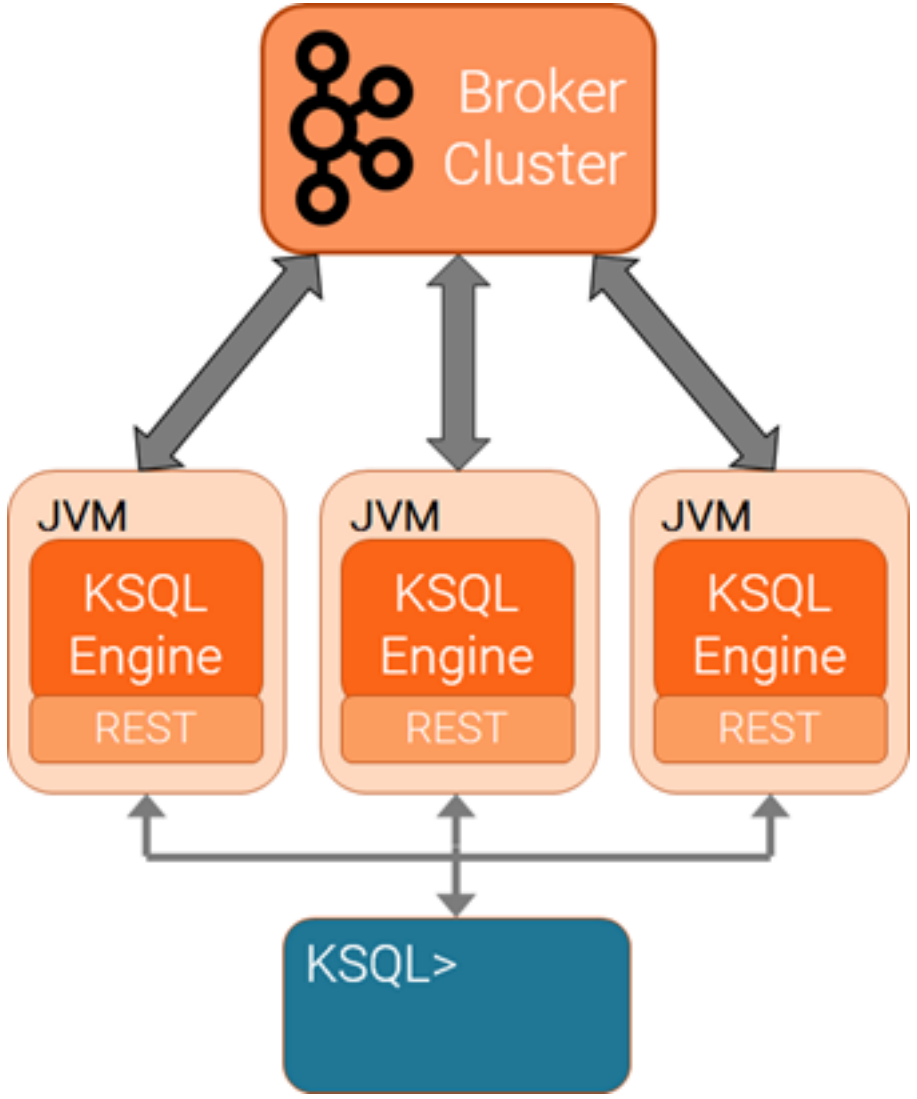
    `WINDOW HOPPING ( size 20 second, advance by 5 second) GROUP BY ip;`
- SESSION: Dynamically-sized, non-overlapping, data-driven window
  - `SELECT ip, SUM(bytes) AS bytes_per_ip FROM clickstream`

    `WINDOW SESSION (20 second) GROUP BY ip;`

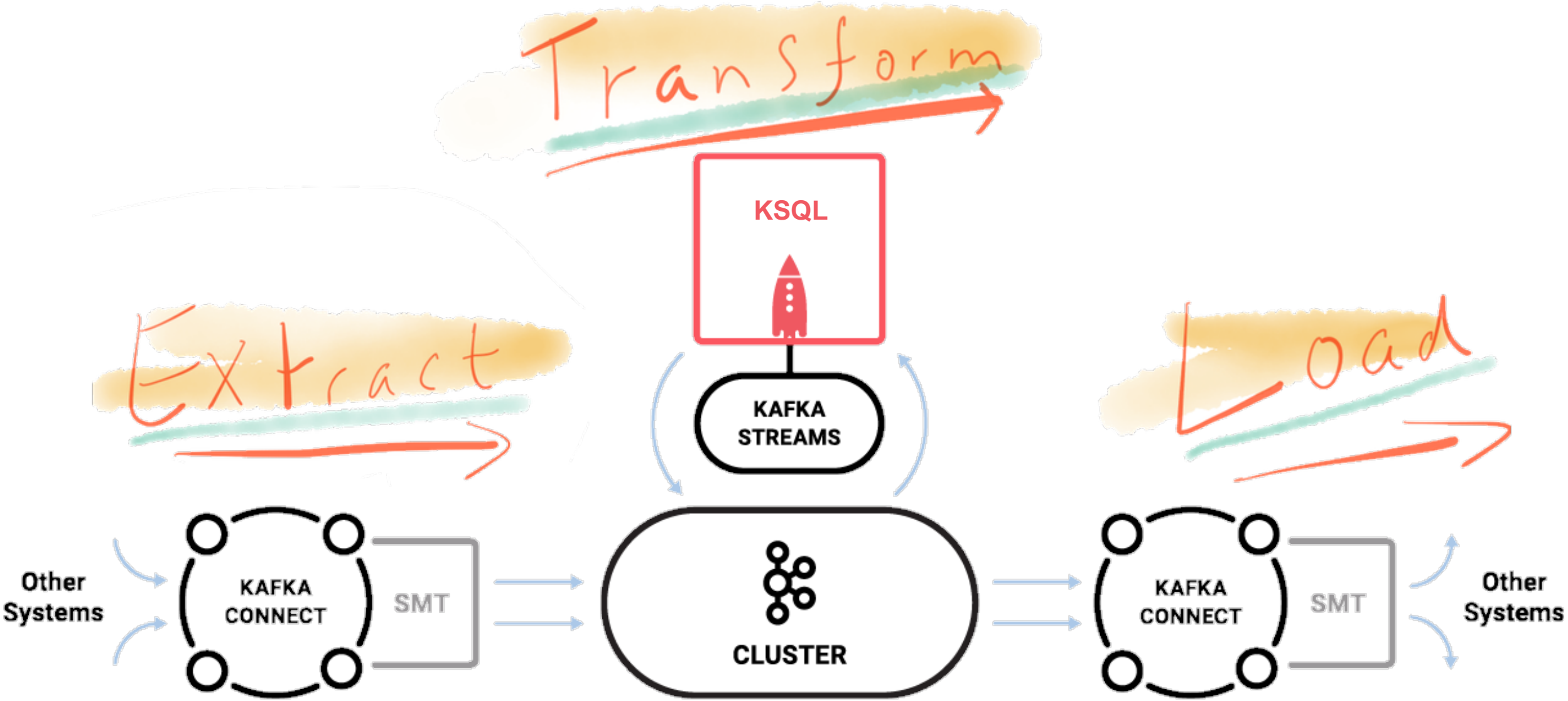
More: http://docs.confluent.io/current/streams/developer-guide.html#windowing

KSQL Deployment Models – Local, or Client/Server

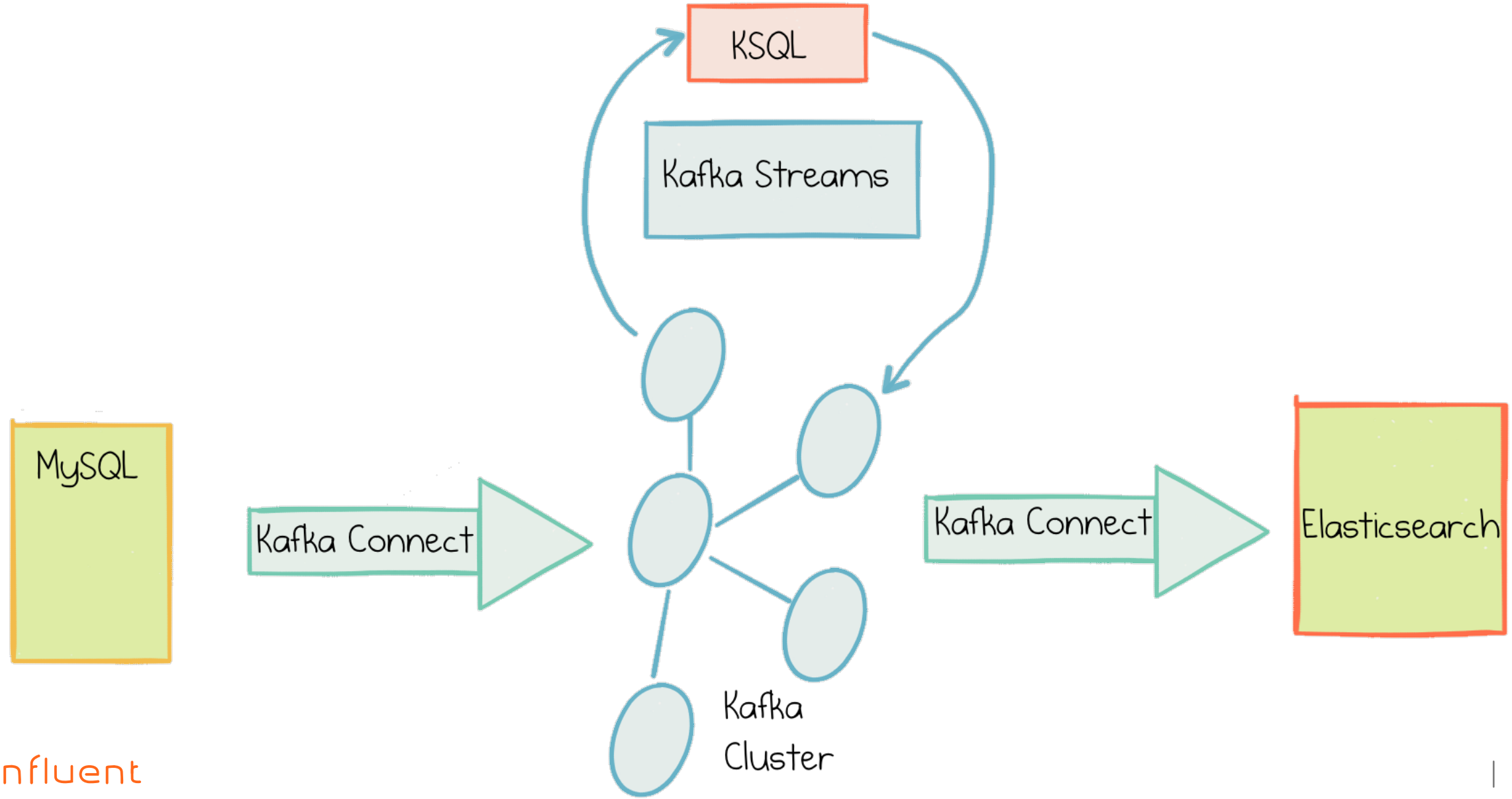




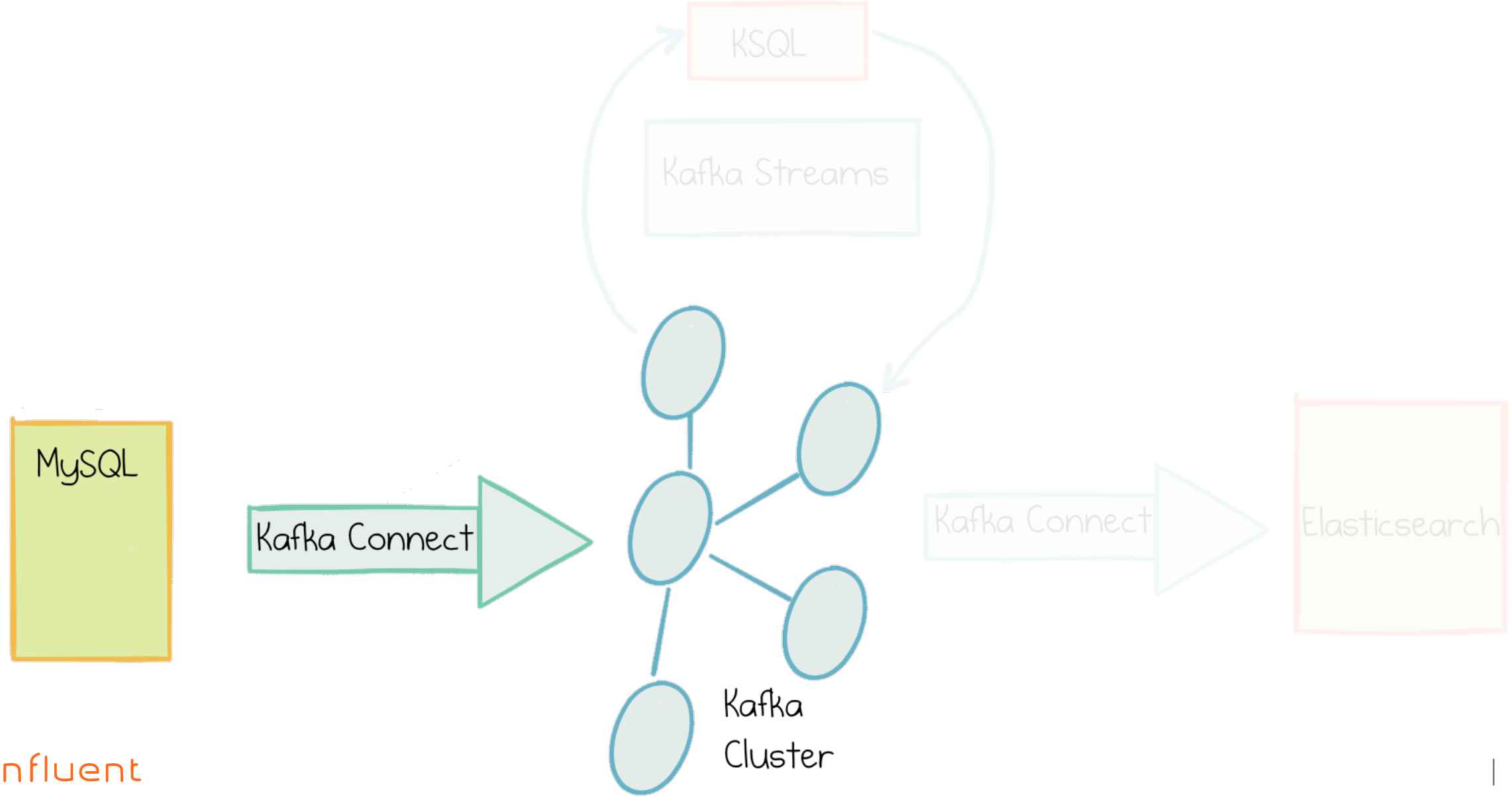
Streaming ETL, powered by Apache Kafka and Confluent Platform



Streaming ETL with Apache Kafka and Confluent Platform



Streaming ETL with Apache Kafka and Confluent Platform



Define a connector

```
{
  "name": "jdbc_source_mysql_01",
  "config": {
    "connector.class": "io.confluent.connect.jdbc.JdbcSourceConnector",
    "connection.url": "jdbc:mysql://localhost:3306/sakila?user=rmoff&password=pw",
    "schema.pattern": "sakila",
    "mode": "bulk",
    "topic.prefix": "sakila-"
  }
}
```

Load the connector

```
Robin@asgard02:~> confluent status
connect is [UP]
kafka-rest is [UP]
schema-registry is [UP]
kafka is [UP]
zookeeper is [UP]
Robin@asgard02:~> confluent load jdbc-source -d /tmp/jdbc.json
Robin@asgard02:~> confluent status jdbc_source_mysql_foobar_01
{
  "name": "jdbc_source_mysql_foobar_01",
  "connector": {
    "state": "RUNNING",
    "worker_id": "127.0.0.1:8083"
  },
```

Tables → Topics

```
mysql> show full tables where Table_Type = 'BASE TABLE';
+------------------+------------+
| Tables_in_sakila | Table_type |
+------------------+------------+
| actor            | BASE TABLE |
| address          | BASE TABLE |
| category         | BASE TABLE |
| city
| country
| customer
```

```
Robin@asgard02 ~> kafka-topics --zookeeper localhost:2181 \
                  --list | \
                  grep sakila
sakila-actor
sakila-address
sakila-category
sakila-city
sakila-country
sakila-customer
```

confluent

Row → Message

```
mysql> select * from customer limit 1;
+-------------+----------+------------+-----------+--------------------------------+
| customer_id | store_id | first_name | last_name | email                          |
+-------------+----------+------------+-----------+--------------------------------+
|           1 |        1 | MARY       | SMITH     | MARY.SMITH@sakilacustomer.org |
+-------------+----------+------------+-----------+--------------------------------+
```

```
Robin@asgard02 ~> kafka-avro-console-consumer \
                  --bootstrap-server localhost:9092 \
                  --from-beginning \
                  --property schema.registry.url=http://localhost:8081 \
                  --max-messages 1 \
                  --topic sakila-customer | \
                  jq '.'
{
  "customer_id": 1,
  "store_id": 1,
  "first_name": "MARY",
  "last_name": "SMITH",
  "email": {
    "string": "MARY.SMITH@sakilacustomer.org"
```

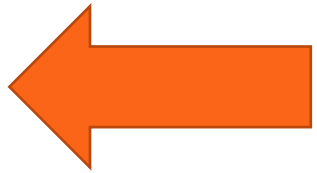
Single Message Transforms

```
"transforms": "InsertTopic,InsertSourceDetails",
"transforms.InsertTopic.type": "org.apache.kafka.connect.transforms.InsertField$Value",
"transforms.InsertTopic.topic.field": "messagetopic",
"transforms.InsertSourceDetails.type": "org.apache.kafka.connect.transforms.InsertField$Value",
"transforms.InsertSourceDetails.static.field": "messagesource",
"transforms.InsertSourceDetails.static.value": "MySQL demo on asgard",
```

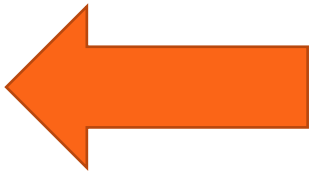
https://www.confluent.io/blog/simplest-useful-kafka-connect-data-pipeline-world-thereabouts-part-3/
http://kafka.apache.org/documentation.html#connect_transforms

Single Message Transforms

```
Robin@asgard02 ~> kafka-console-consumer --boot
c sakila-customer | jq '.'
{
  "customer_id": 1,
  "store_id": 1,
  "first_name": "MARY",
  "last_name": "SMITH",
  "email": "MARY.SMITH@sakilacustomer.org",
  "address_id": 5,
  "active": 1,
  "create_date": 1139954676000,
  "last_update": 1139979440000,
  "messagetopic": "sakila-customer",
  "messagesource": "MySQL demo on asgard"
}
```



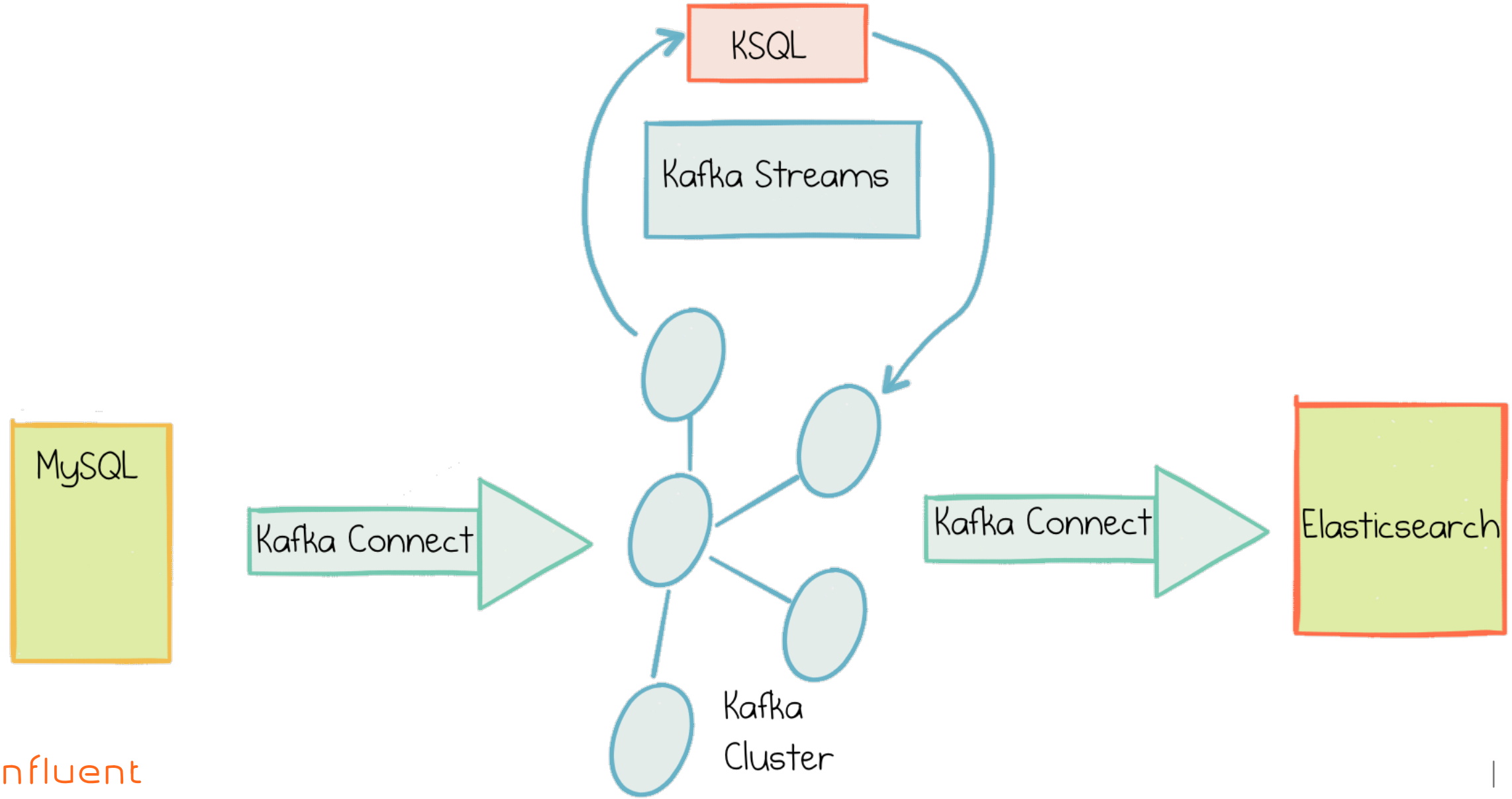
Record data



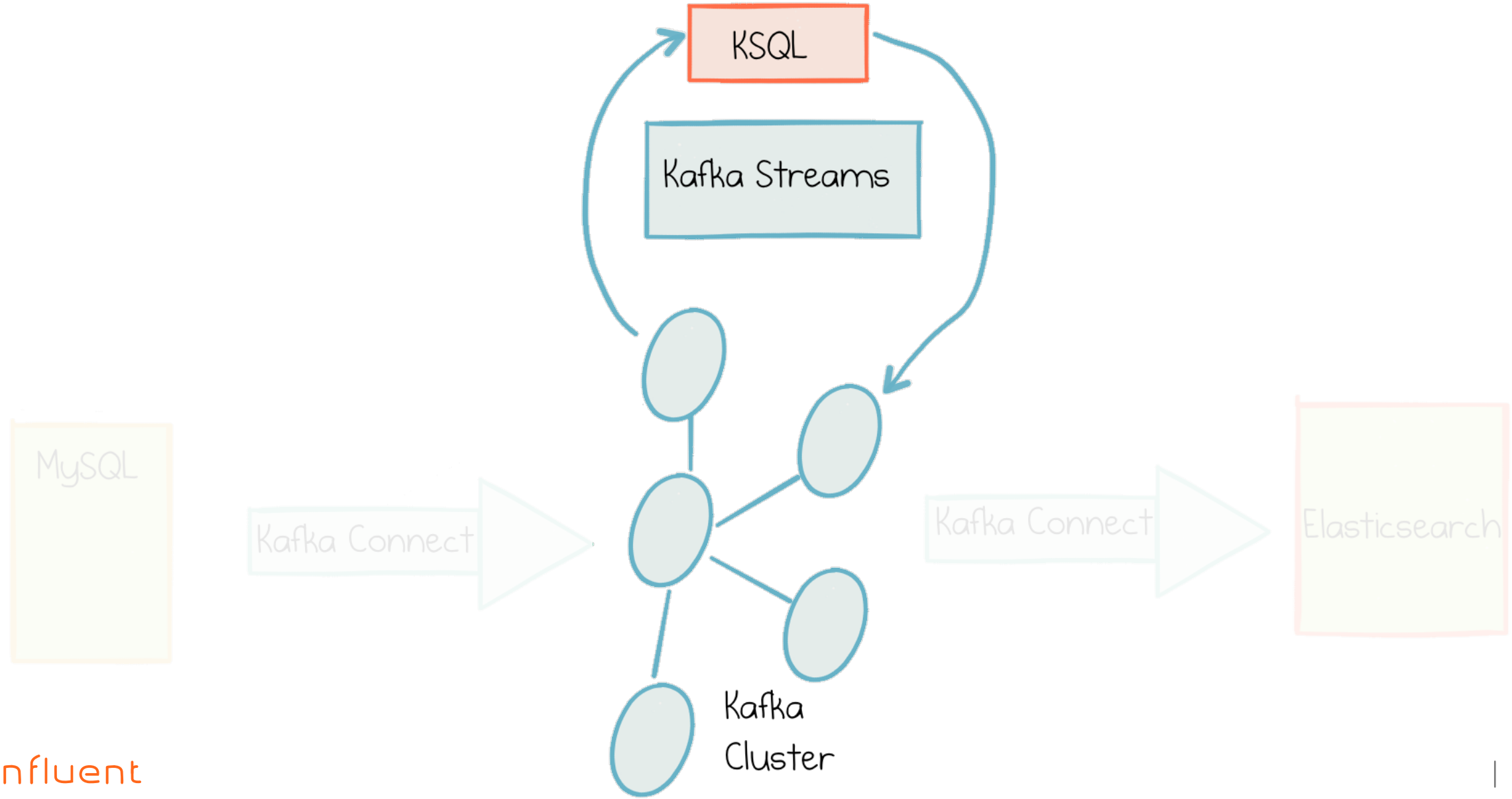
Bespoke lineage data

https://www.confluent.io/blog/simplest-useful-kafka-connect-data-pipeline-world-thereabouts-part-3/
http://kafka.apache.org/documentation.html#connect_transforms

Streaming ETL with Apache Kafka and Confluent Platform



Streaming ETL with Apache Kafka and Confluent Platform



KSQL in action

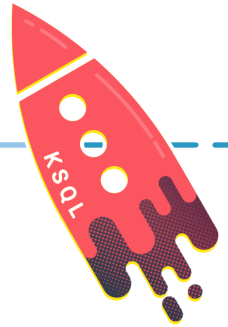
```
ksql> CREATE stream rental
      (rental_id INT, rental_date INT, inventory_id INT,
       customer_id INT, return_date INT, staff_id INT,
      last_update INT )
      WITH (kafka_topic = 'sakila-rental',
      value_format = 'json');

Message
---------------
Stream created
```

confluent

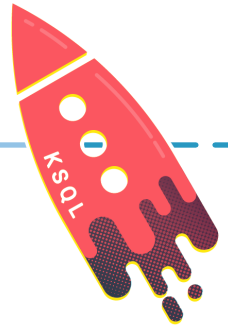
* Command formatted for clarity here.
Linebreaks need to be denoted by \ in KSQL

KSQL in action



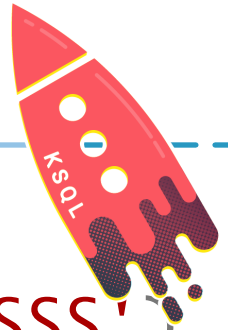

```
ksql> describe rental;
Field          | Type
-------------------------------
ROWTIME        | BIGINT
ROWKEY         | VARCHAR(STRING)
RENTAL_ID      | INTEGER
RENTAL_DATE    | INTEGER
INVENTORY_ID   | INTEGER
CUSTOMER_ID    | INTEGER
RETURN_DATE    | INTEGER
STAFF_ID       | INTEGER
LAST_UPDATE    | INTEGER
```

KSQL in action




```
ksql> select * from rental limit 3;
1505830937567 | null | 1 | 280113040 | 367 | 130 |
1505830937567 | null | 2 | 280176040 | 1525 | 459 |
1505830937569 | null | 3 | 280722040 | 1711 | 408 |
```

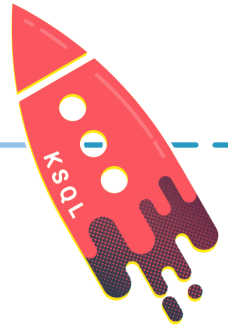
KSQL in action




```
SELECT rental_id ,
     TIMESTAMPTOSTRING(rental_date, 'yyyy-MM-dd HH:mm:ss.SSS'),
     TIMESTAMPTOSTRING(return_date, 'yyyy-MM-dd HH:mm:ss.SSS')
FROM rental
limit 3;

 1 | 2005-05-24 22:53:30.000 | 2005-05-26 22:04:30.000
 2 | 2005-05-24 22:54:33.000 | 2005-05-28 19:40:33.000
 3 | 2005-05-24 23:03:39.000 | 2005-06-01 22:12:39.000
 LIMIT reached for the partition.
 Query terminated
 ksql>
```

KSQL in action




```
SELECT rental_id ,
       TIMESTAMPTOSTRING(rental_date, 'yyyy-MM-dd HH:mm:ss.SSS'),
       TIMESTAMPTOSTRING(return_date, 'yyyy-MM-dd HH:mm:ss.SSS'),
       ceil((cast(return_date AS DOUBLE) -
             cast(rental_date AS DOUBLE) )
            / 60 / 60 / 24 / 1000)
FROM rental;
```

```
1 | 2005-05-24 22:53:30.000 | 2005-05-26 22:04:30.000 | 2.0
2 | 2005-05-24 22:54:33.000 | 2005-05-28 19:40:33.000 | 4.0
3 | 2005-05-24 23:03:39.000 | 2005-06-01 22:12:39.000 | 8.0
```

KSQL in action

```
CREATE stream rental_lengths AS
SELECT rental_id ,
TIMESTAMPTOSTRING(rental_date, 'yyyy-MM-dd HH:mm:ss.SSS') ,
TIMESTAMPTOSTRING(return_date, 'yyyy-MM-dd HH:mm:ss.SSS') ,
ceil(( cast(return_date AS DOUBLE) -  cast( rental_date AS DOUBLE)
) / 60 / 60 / 24 / 1000)
FROM rental;
```

KSQL in action

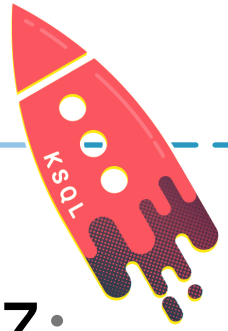
```
ksql> select rental_id, rental_date, return_date,
RENTAL_LENGTH_DAYS from rental_lengths;
3 | 2005-05-24 23:03:39.000 | 2005-06-01 22:12:39.000 | 8.0
4 | 2005-05-24 23:04:41.000 | 2005-06-03 01:43:41.000 | 10.0
7 | 2005-05-24 23:11:53.000 | 2005-05-29 20:34:53.000 | 5.0
```

KSQL in action

```
$ kafka-topics --zookeeper localhost:2181 --list
RENTAL_LENGTHS

$ kafka-console-consumer --bootstrap-server localhost:9092
--from-beginning --topic RENTAL_LENGTHS | jq '.'
{  "RENTAL_DATE": "2005-05-24 22:53:30.000",
   "RENTAL_LENGTH_DAYS": 2,
   "RETURN_DATE": "2005-05-26 22:04:30.000",
   "RENTAL_ID": 1
}
```

KSQL in action



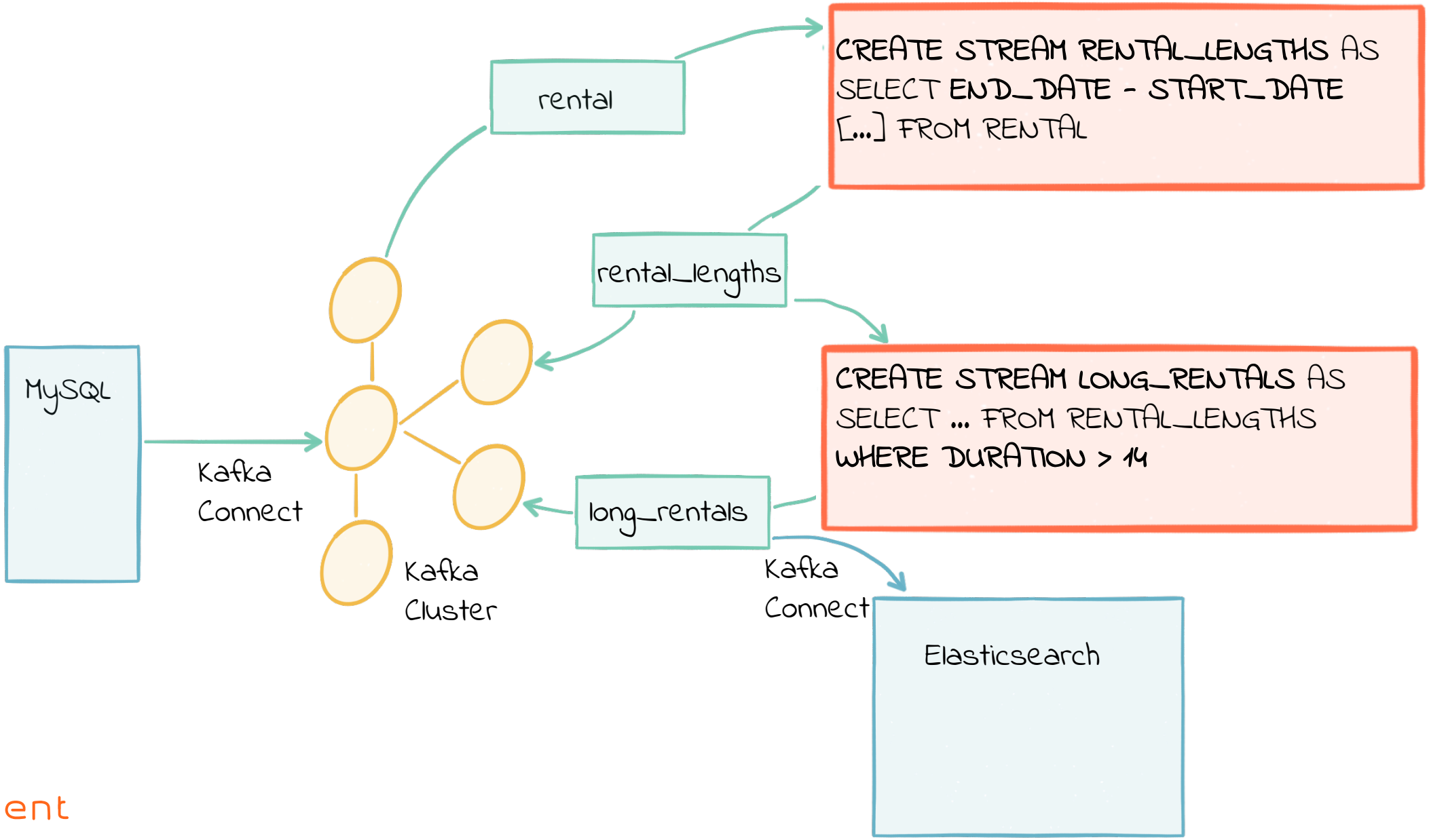

```
CREATE stream long_rentals AS
SELECT * FROM rental_lengths WHERE rental_length_days > 7;


ksql> select rental_id, rental_date, return_date,
RENTAL_LENGTH_DAYS from long_rentals;
3 | 2005-05-24 23:03:39.000 | 2005-06-01 22:12:39.000 | 8.0
4 | 2005-05-24 23:04:41.000 | 2005-06-03 01:43:41.000 | 10.0
```

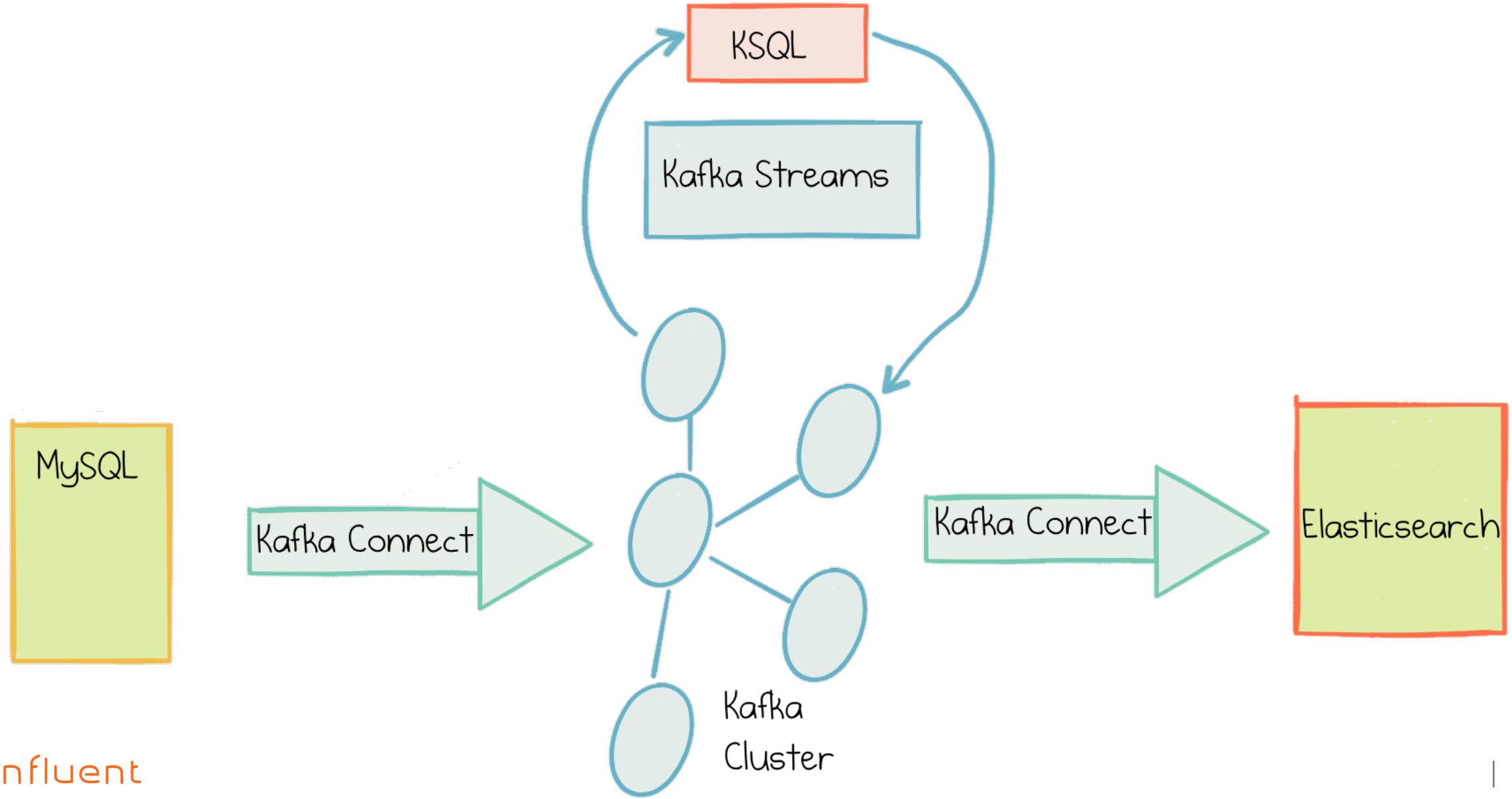
KSQL in action

```
$ kafka-console-consumer --bootstrap-server localhost:9092
--from-beginning --topic LONG_RENTALS | jq '.'
{  "RENTAL_DATE": " 2005-05-24 23:03:39.000",
   "RENTAL_LENGTH_DAYS": 8,
   "RETURN_DATE": " 2005-06-01 22:12:39.000",
   "RENTAL_ID": 3
}
```

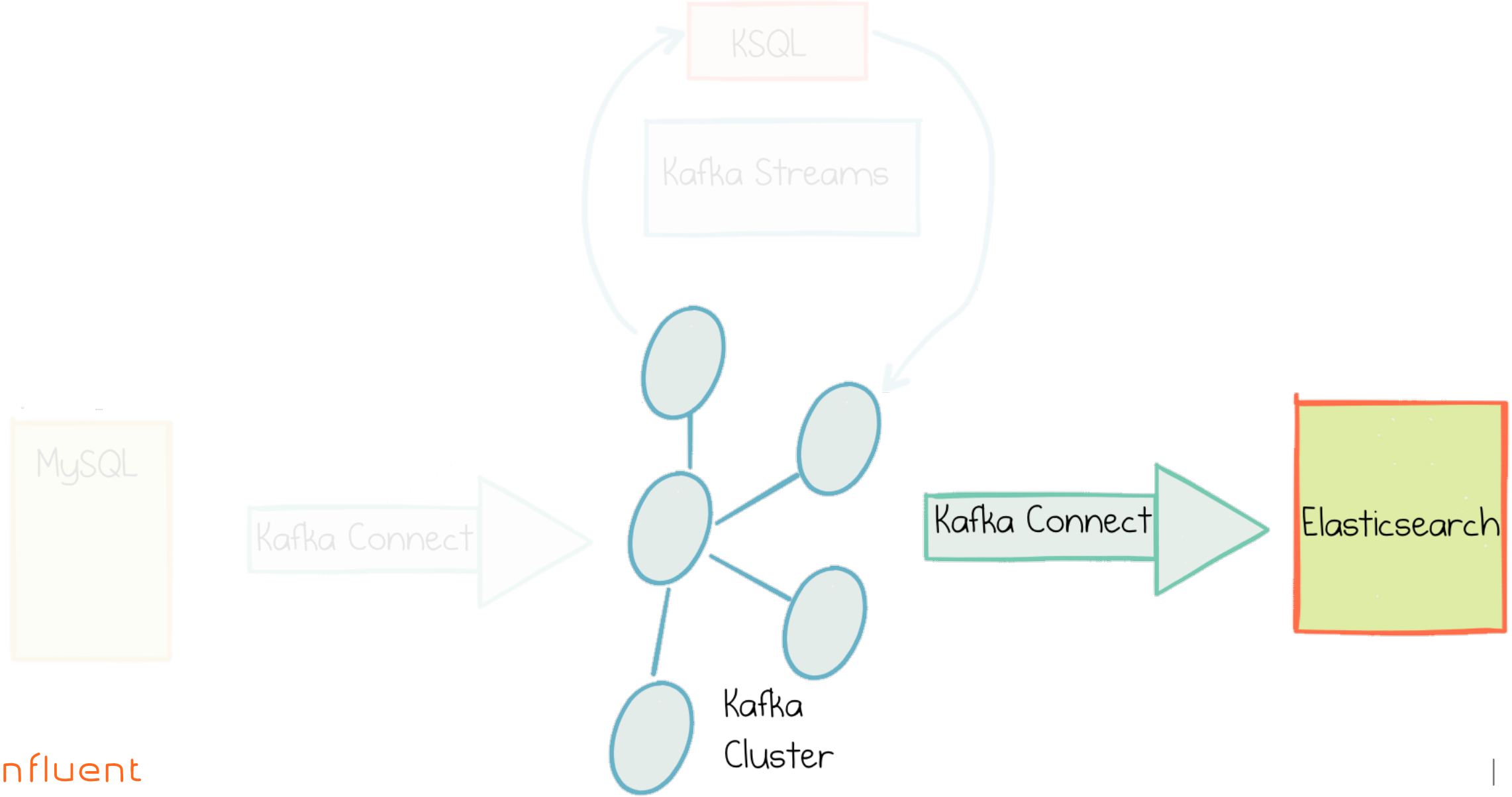
Streaming ETL with Kafka Connect and KSQL



Streaming ETL with Apache Kafka and Confluent Platform



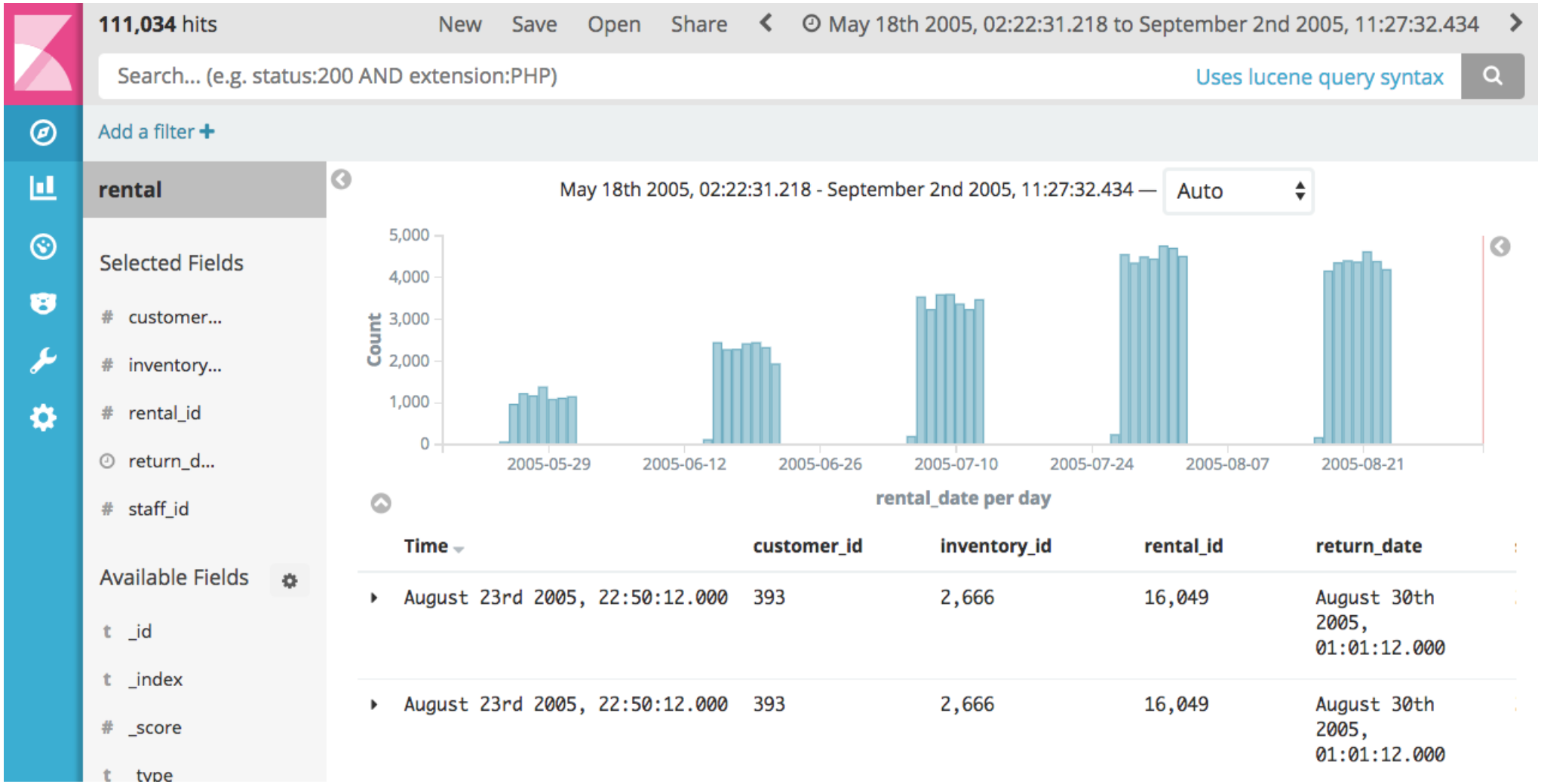
Streaming ETL with Apache Kafka and Confluent Platform



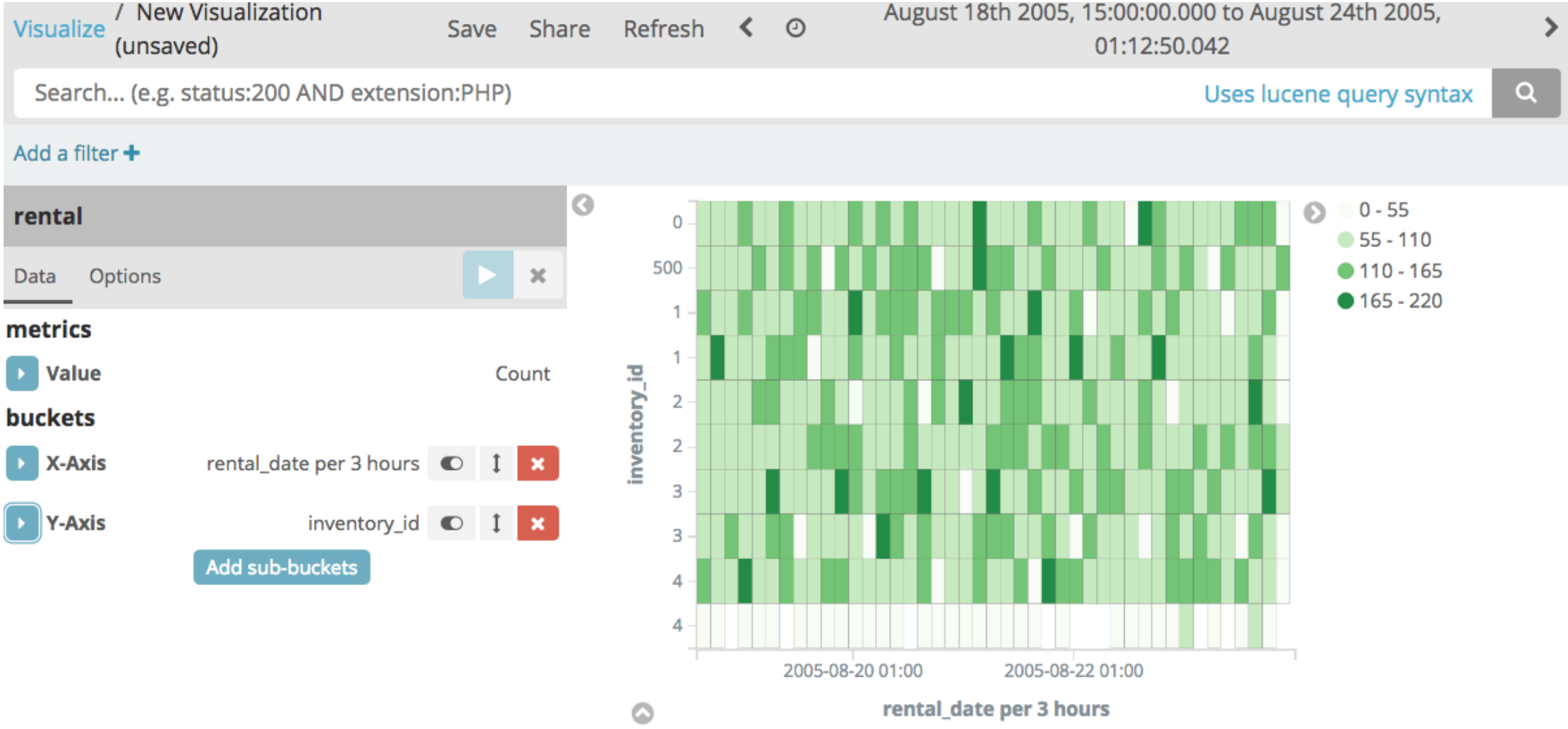
Kafka Connect to stream Kafka Topics to Elasticsearch…MySQL…& more

```
{
  "name": "es-sink-avro-02",
  "config": {
        "connector.class":
          "io.confluent.connect.elasticsearch.ElasticsearchSinkConnector",
        "connection.url": "http://localhost:9200",
        "type.name": "type.name=kafka-connect",
        "topics": "sakila-avro-rental",
        "key.ignore": "true",
        "transforms":"dropPrefix",
        "transforms.dropPrefix.type":"org.apache.kafka.connect.transforms.RegexRouter",
        "transforms.dropPrefix.regex":"sakila-avro-(.*)",
        "transforms.dropPrefix.replacement":"$1"
        }
}
```

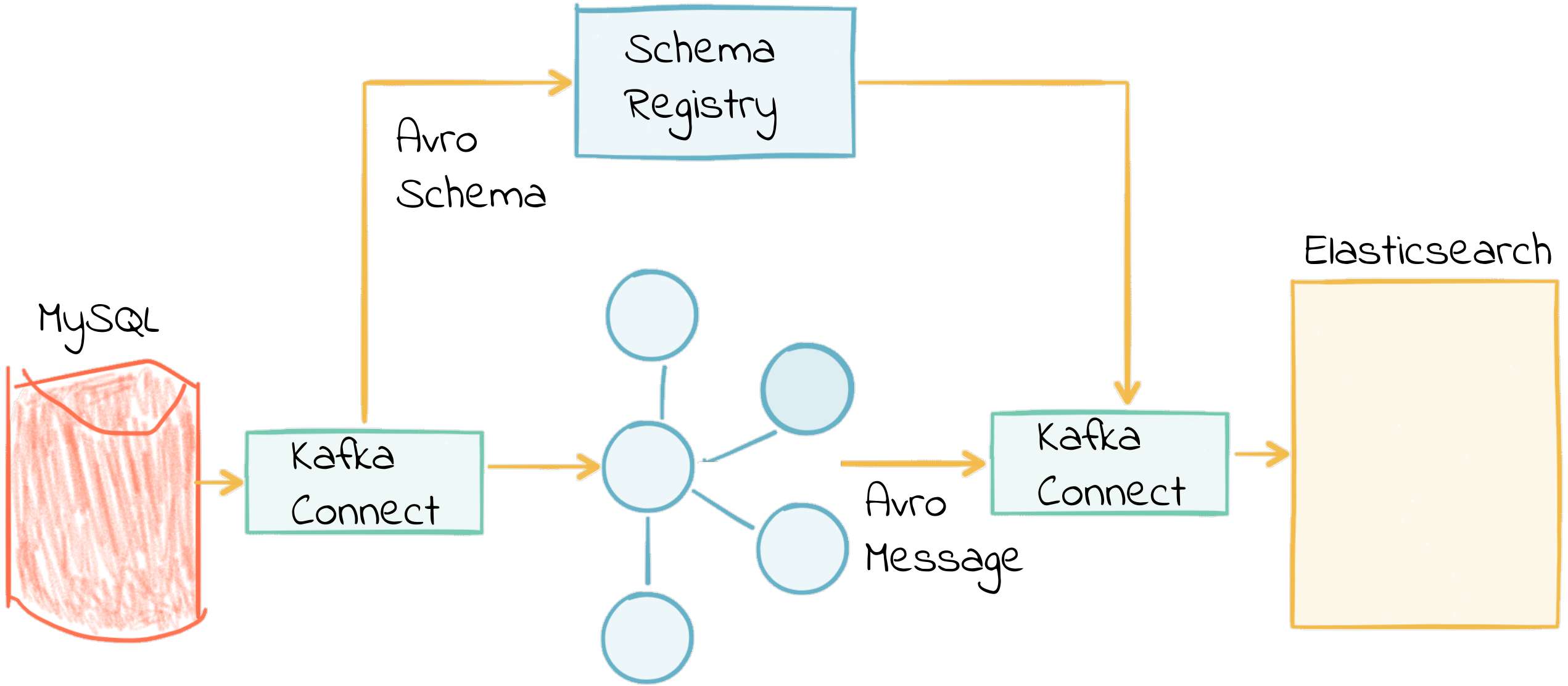
Kafka Connect to stream Kafka Topics to Elasticsearch…MySQL…& more



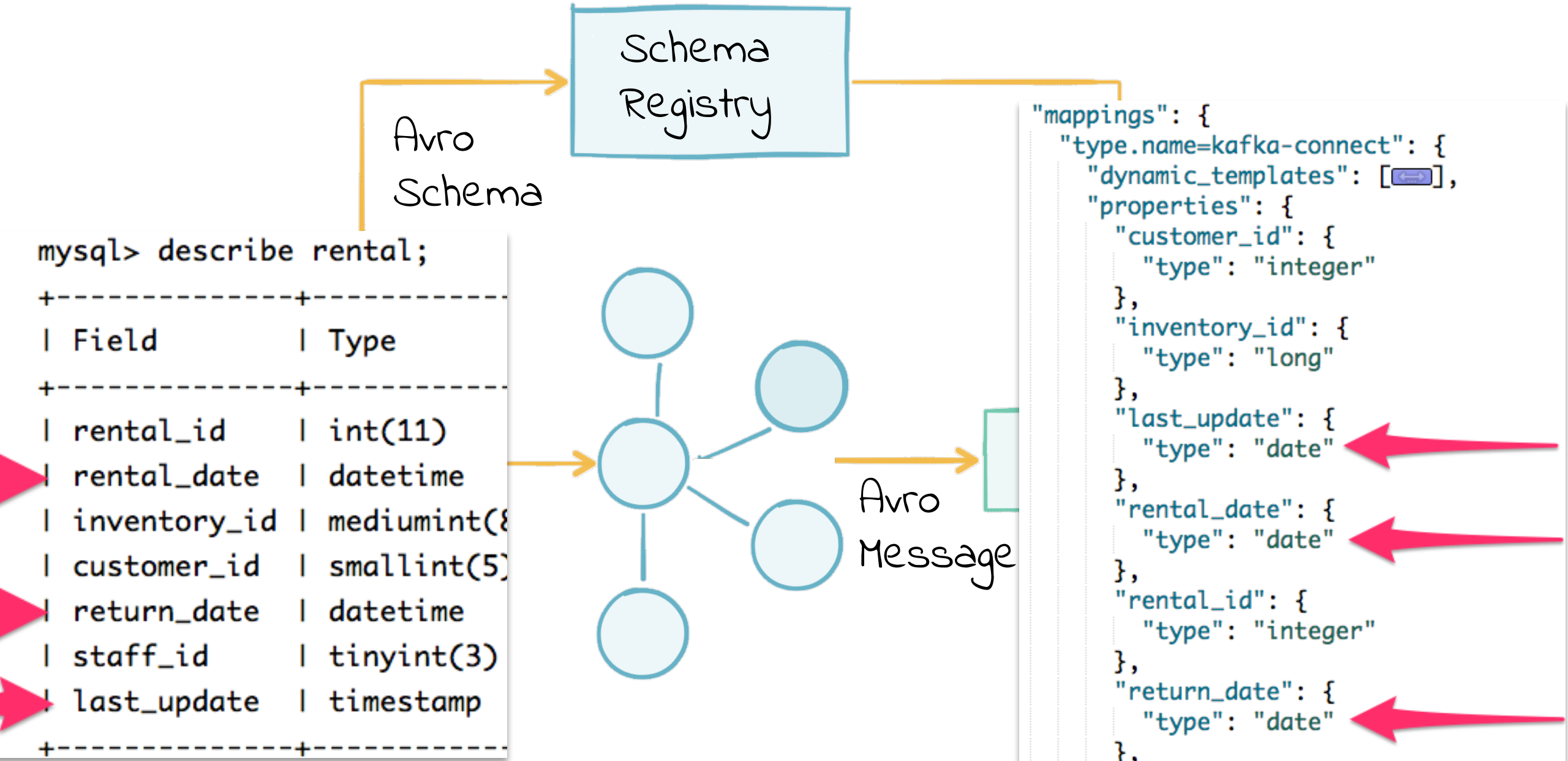
Popular Rental Titles over Time



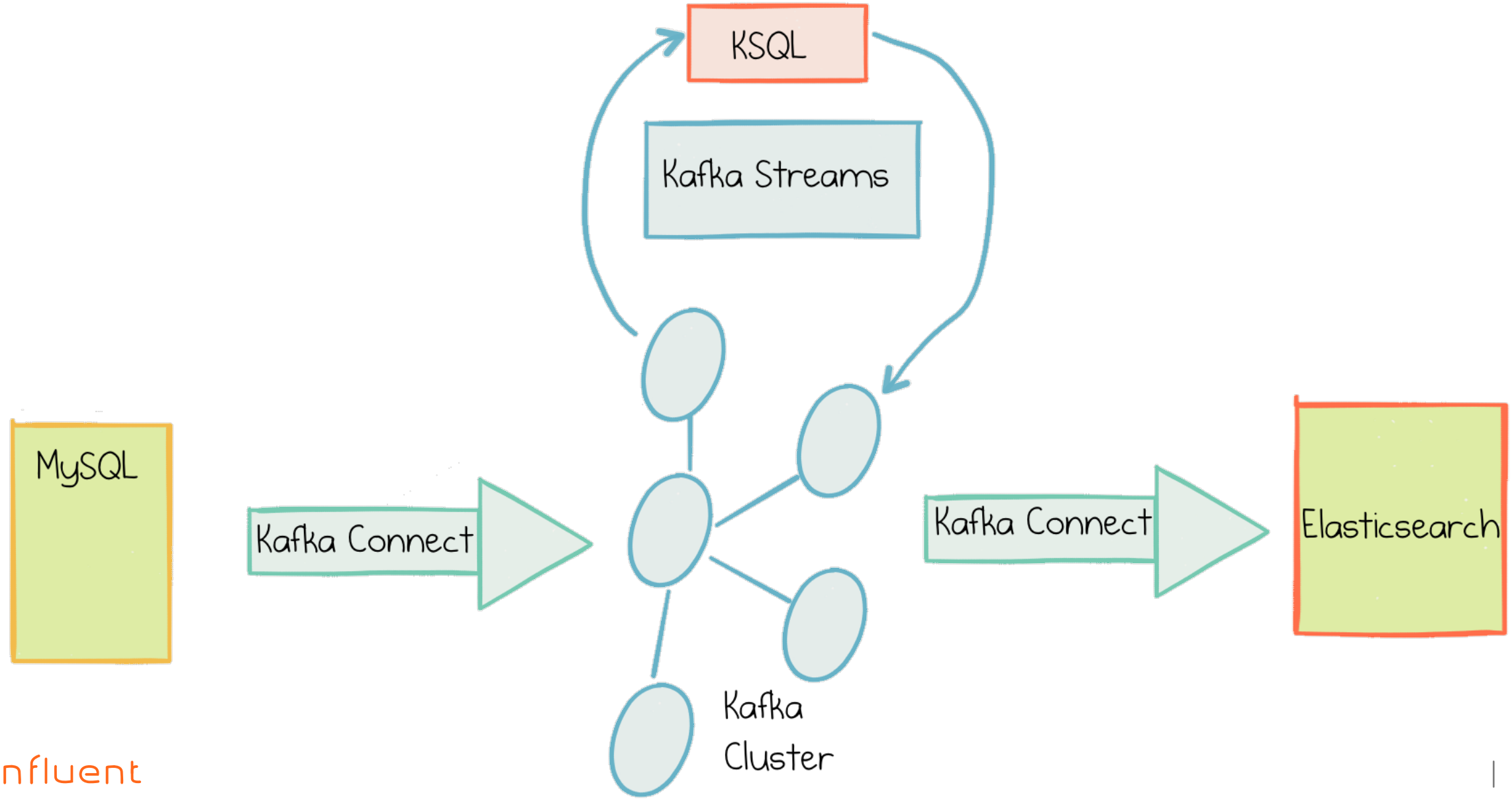
Kafka Connect + Schema Registry = WIN



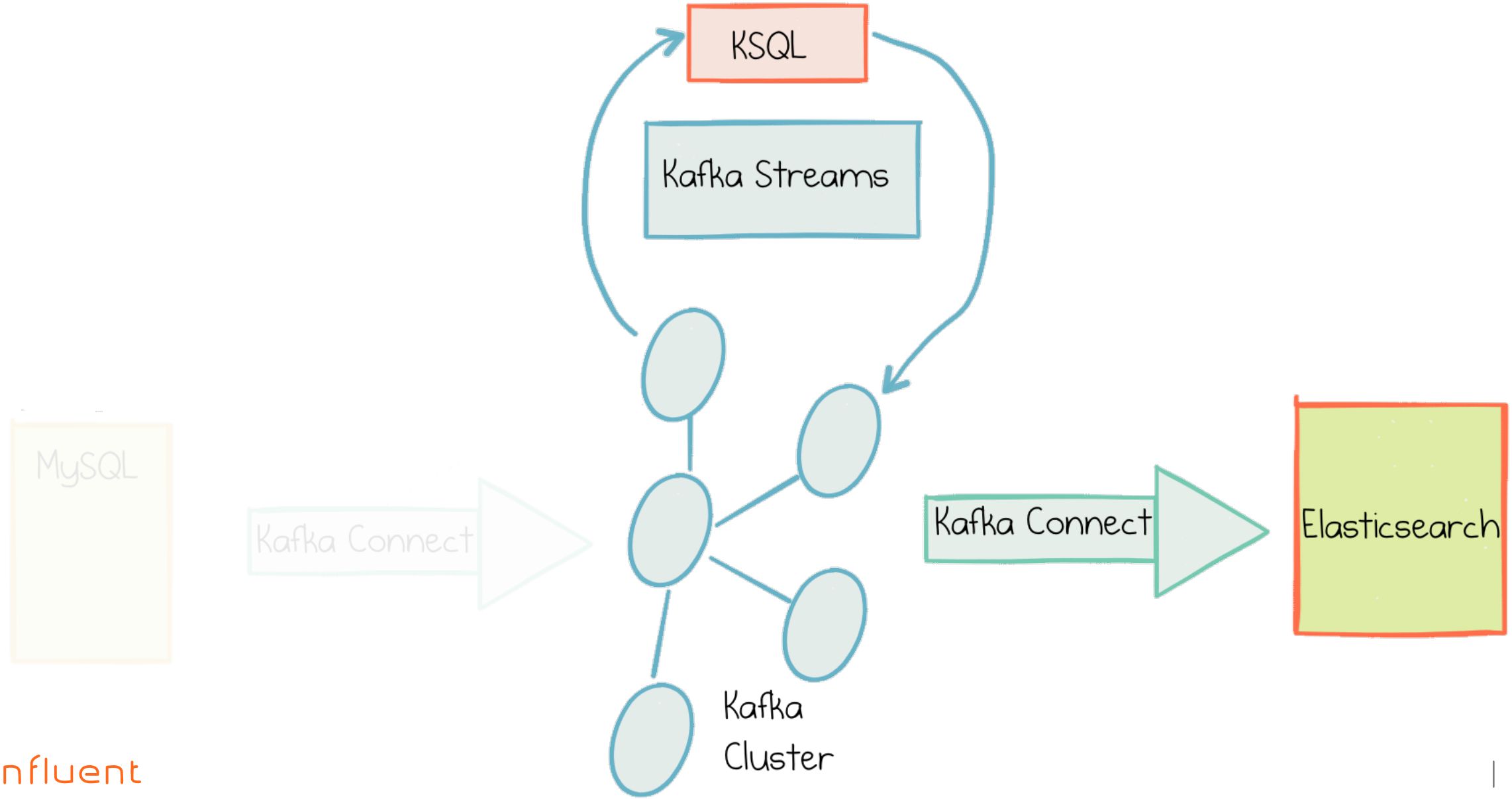
Kafka Connect + Schema Registry = WIN



Streaming ETL with Apache Kafka and Confluent Platform



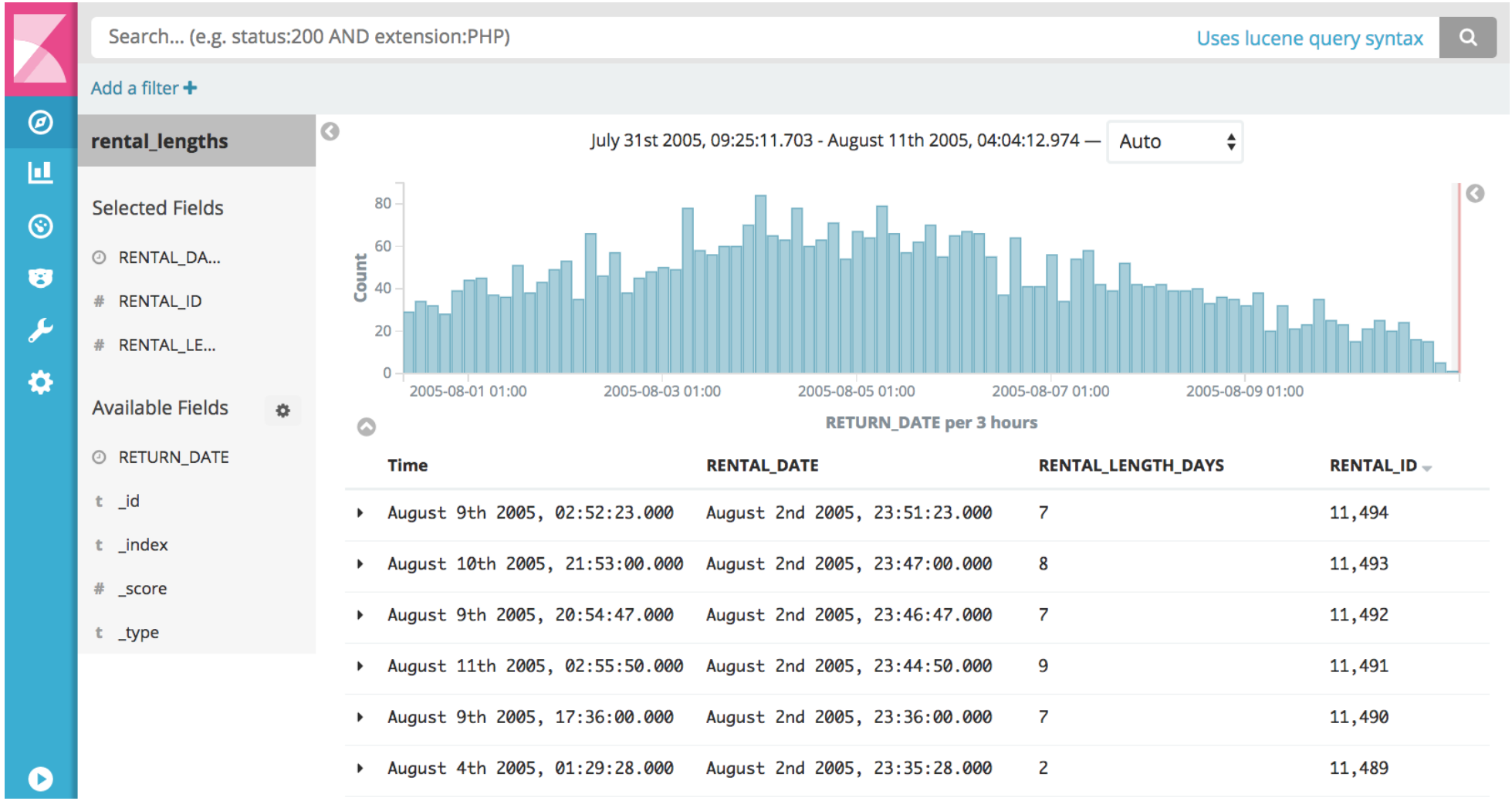
Streaming ETL with Apache Kafka and Confluent Platform



Kafka Connect to stream Kafka Topics to Elasticsearch…MySQL…& more

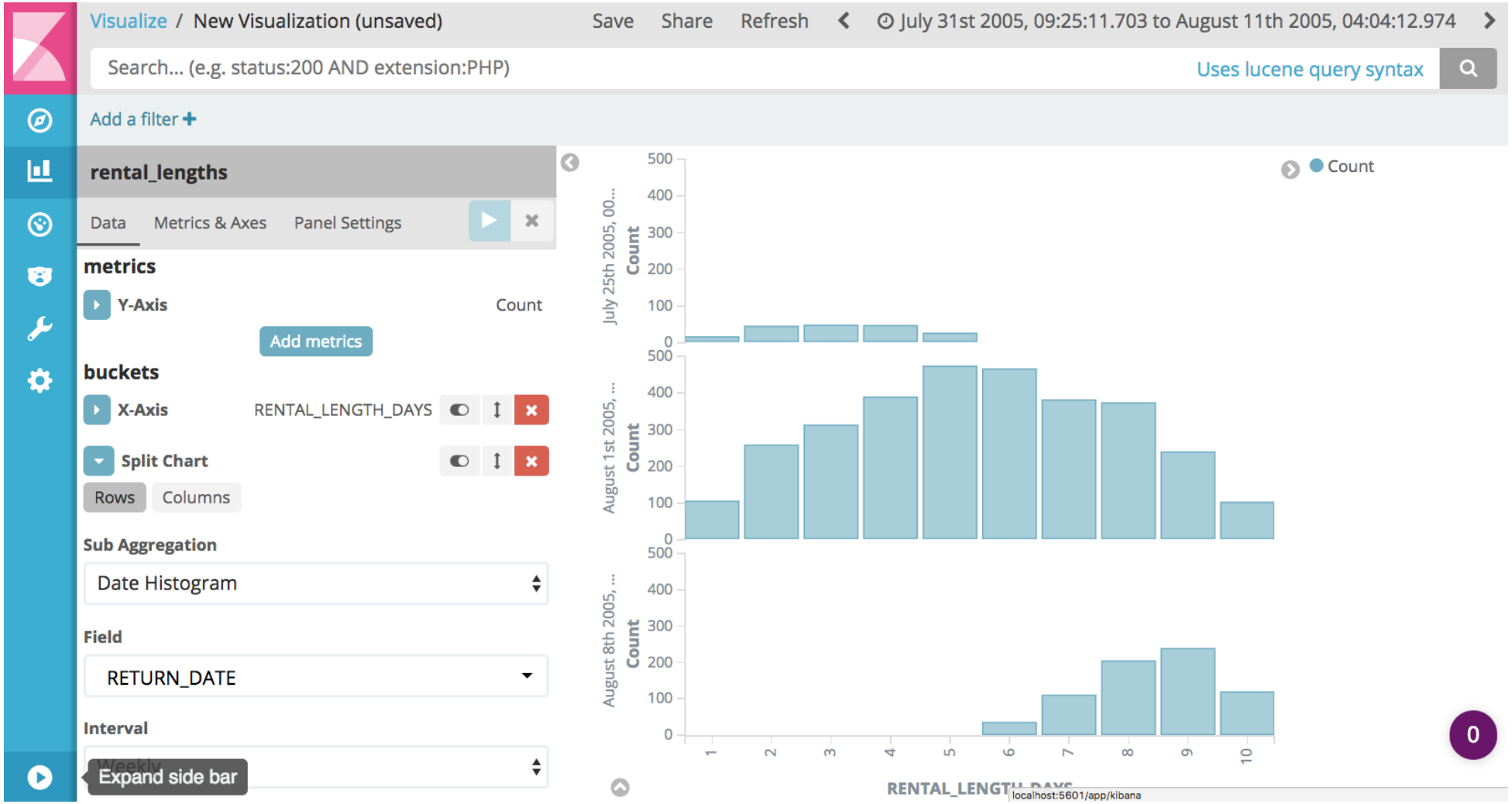
```
{
  "name": "es-sink-rental-lengths-02",
  "config": {
    "connector.class":
    "io.confluent.connect.elasticsearch.ElasticsearchSinkConnector",
    "key.converter": "org.apache.kafka.connect.json.JsonConverter",
    "value.converter": "org.apache.kafka.connect.json.JsonConverter",
    "key.converter.schemas.enable": "false",
    "value.converter.schemas.enable": "false",
    "schema.ignore": "true",
    "connection.url": "http://localhost:9200",
    "type.name": "type.name=kafka-connect",
    "topics": "RENTAL_LENGTHS",
    "topic.index.map": "RENTAL_LENGTHS:rental_lengths",
    "key.ignore": "true"
  }
}
```

Plot data from KSQL-derived stream

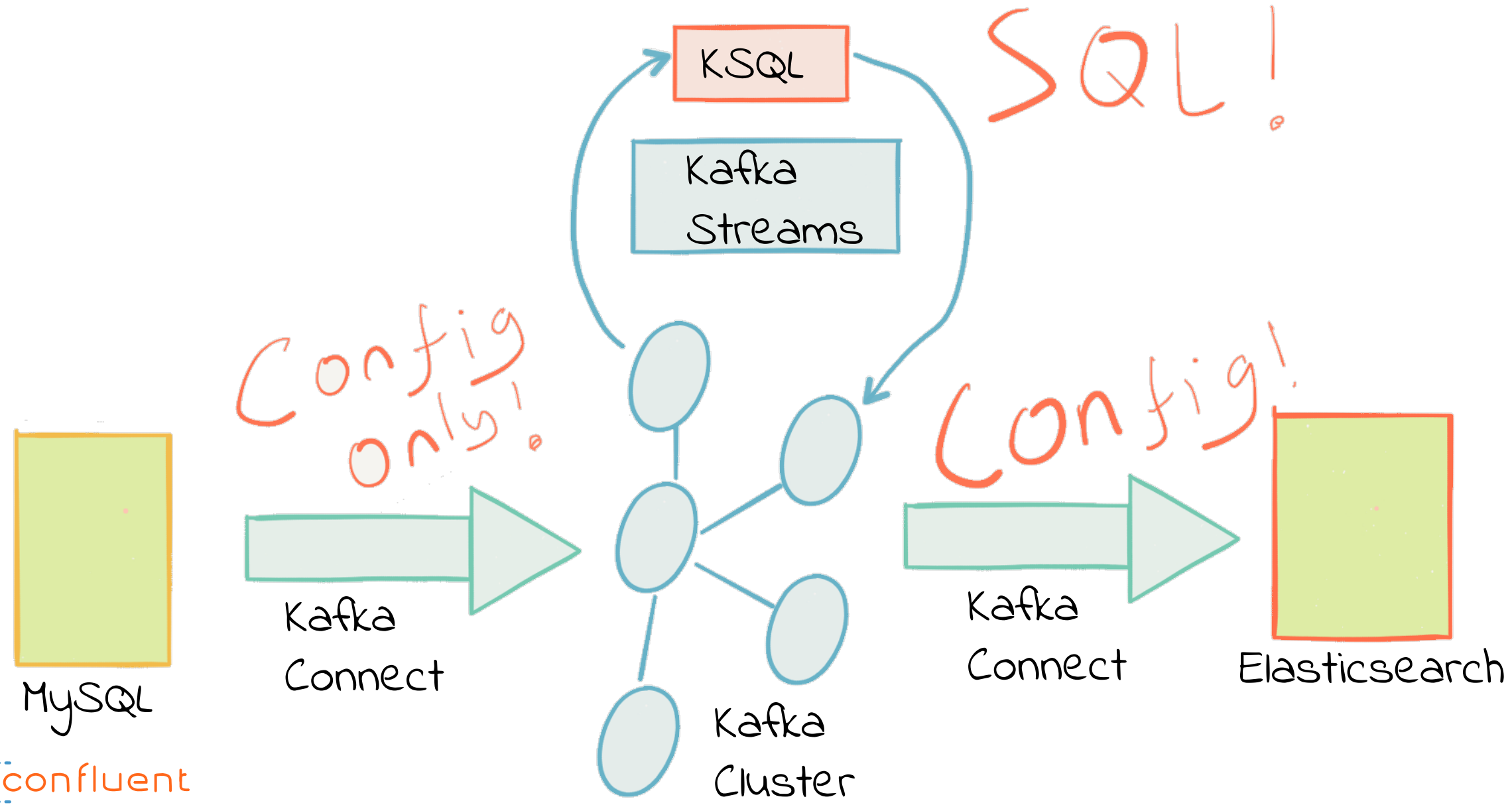


Time	RENTAL_DATE	RENTAL_LENGTH_DAYS	RENTAL_ID
August 9th 2005, 02:52:23.000	August 2nd 2005, 23:51:23.000	7	11,494
August 10th 2005, 21:53:00.000	August 2nd 2005, 23:47:00.000	8	11,493
August 9th 2005, 20:54:47.000	August 2nd 2005, 23:46:47.000	7	11,492
August 11th 2005, 02:55:50.000	August 2nd 2005, 23:44:50.000	9	11,491
August 9th 2005, 17:36:00.000	August 2nd 2005, 23:36:00.000	7	11,490
August 4th 2005, 01:29:28.000	August 2nd 2005, 23:35:28.000	2	11,489

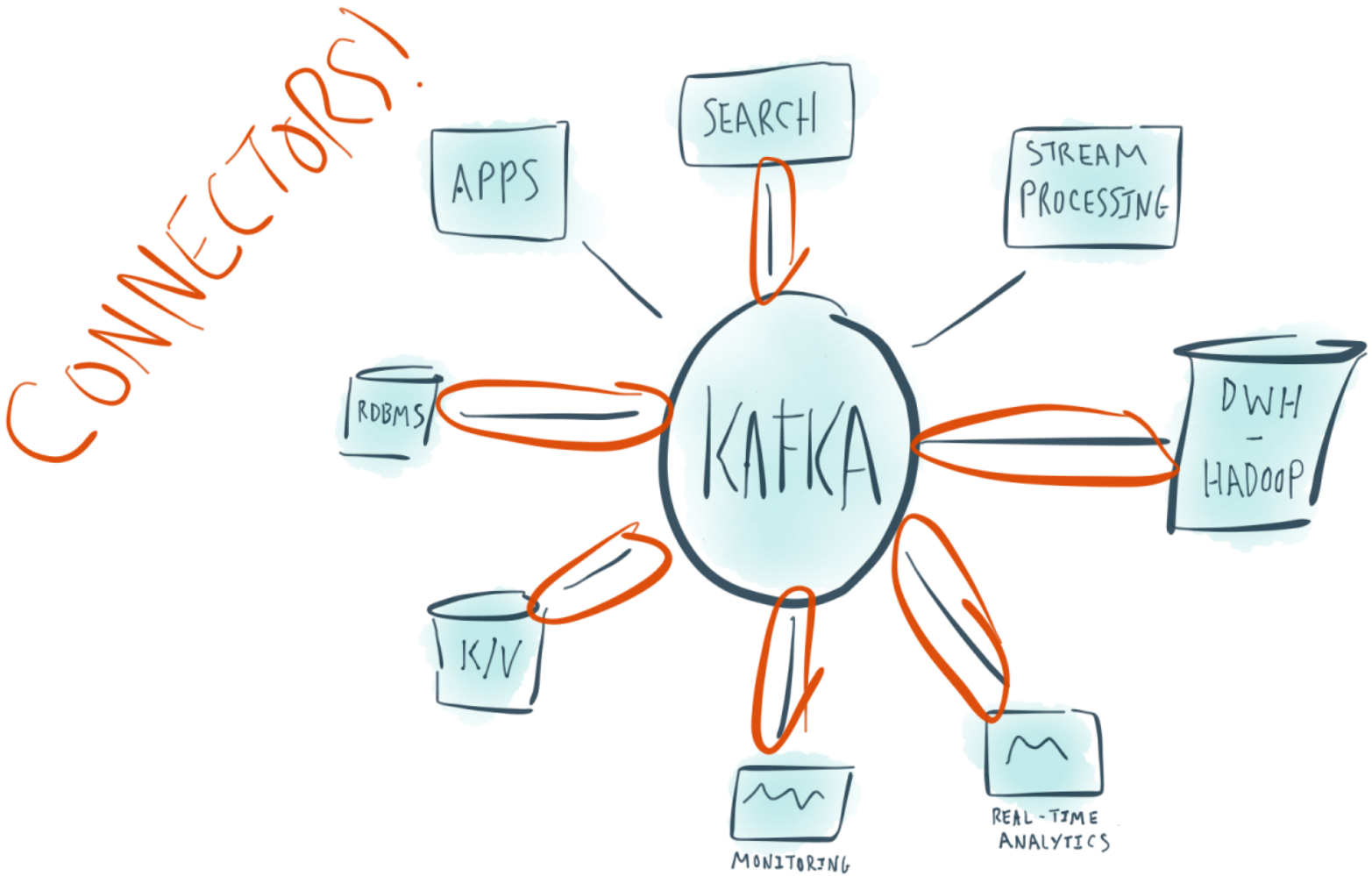
Distribution of rental durations, per week



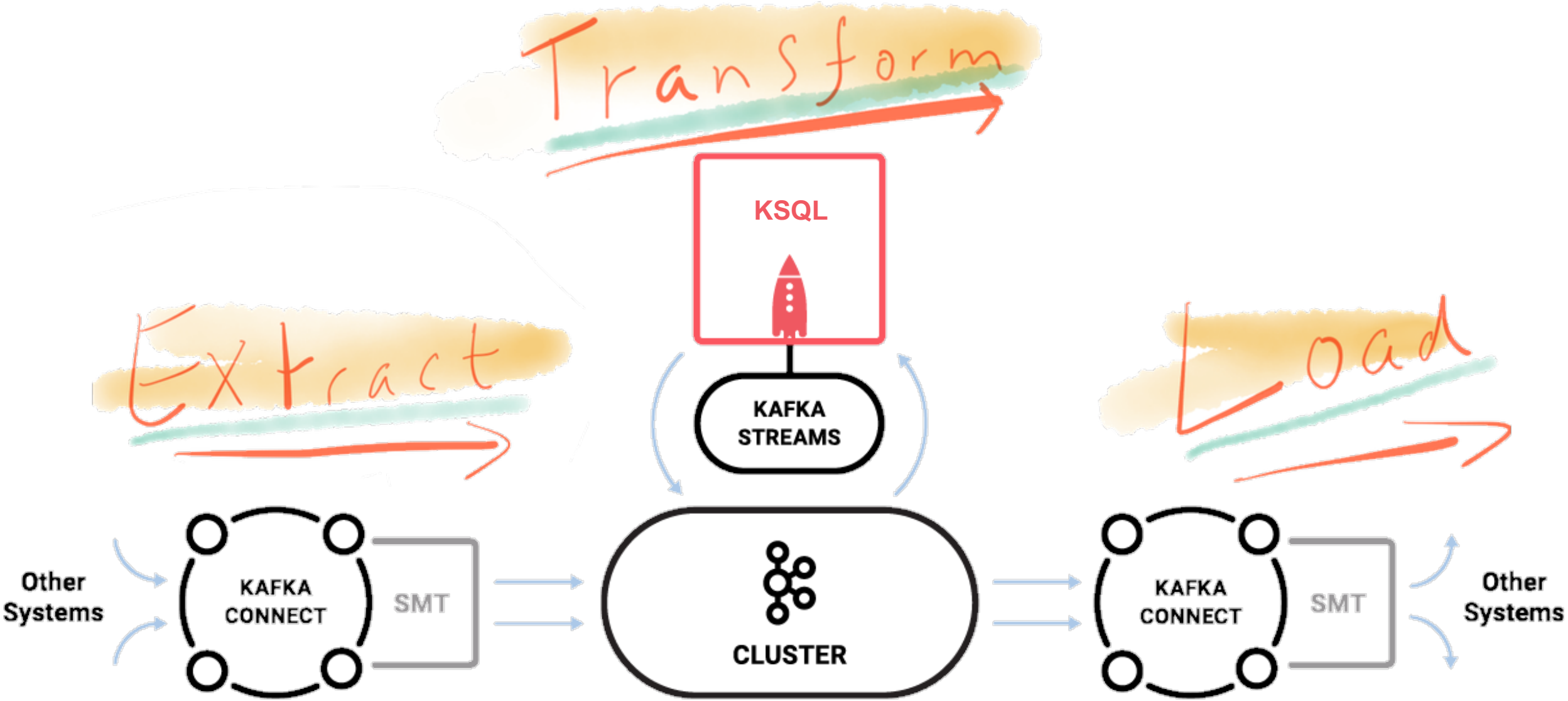
Streaming ETL with Apache Kafka and Confluent Platform – no coding!



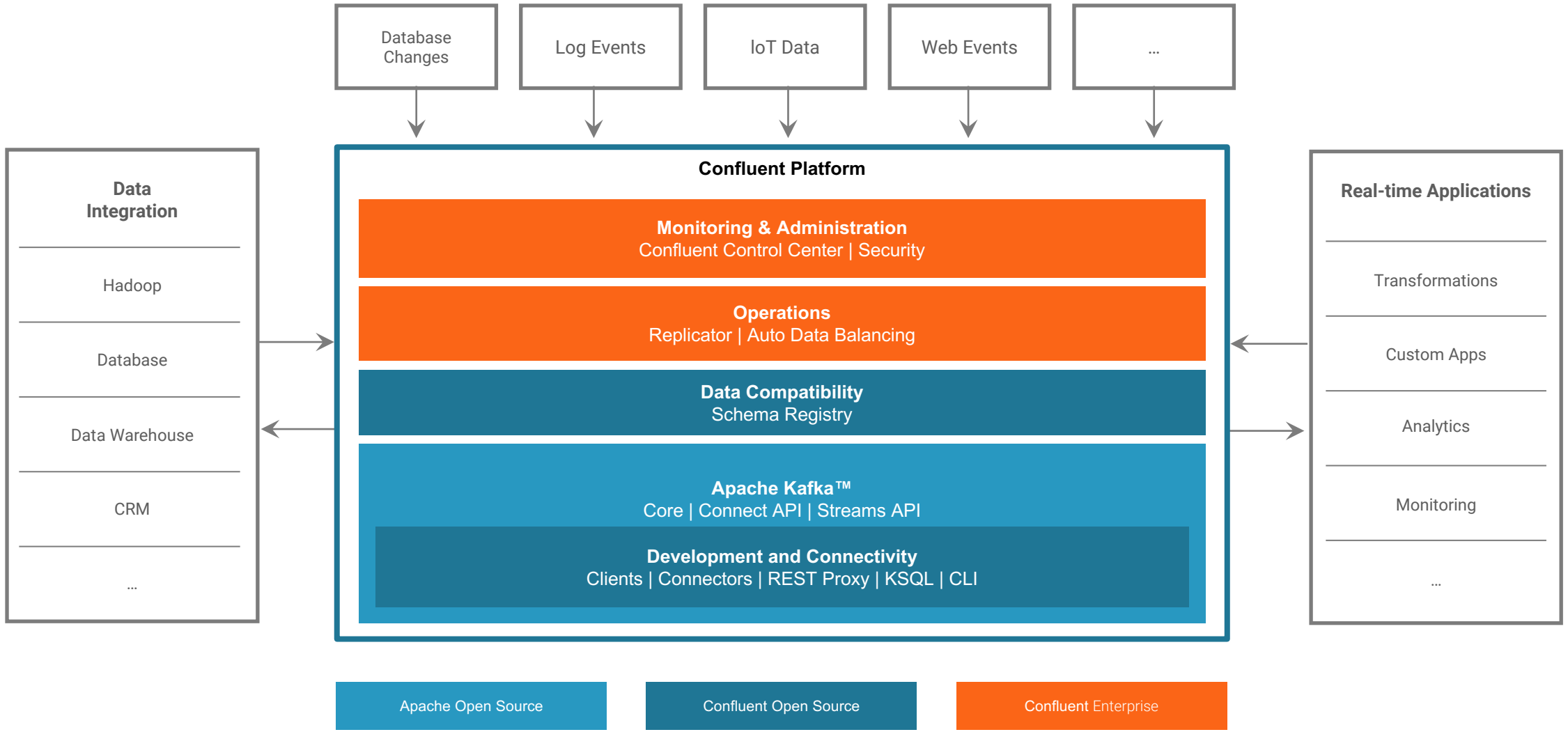
STREAMING PLATFORM



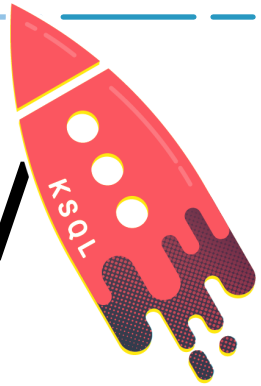
Streaming ETL, powered by Apache Kafka and Confluent Platform



Confluent Platform: Enterprise Streaming based on Apache Kafka™



kafka summit
Kafka Summit 2018
Kafka Summit London: April 23-24, 2018
Kafka Summit San Francisco: October 16-17, 2018
www.kafka-summit.org
Presented by confluent



https://github.com/confluentinc/ksql/

https://www.confluent.io/download/

@gwenshap gwen@confluent.io
@rmoff robin@confluent.io